FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

iLearnTest: Jogo Educativo para Aprendizagem de Testes de Software

Tânia Patrícia Bernardes Ribeiro

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Ana Cristina Ramada Paiva

23 de Julho de 2014

iLearnTest: Jogo Educativo para Aprendizagem de Testes de Software

Tânia Patrícia Bernardes Ribeiro

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: José Manuel Magalhães Cruz

Arguente: José Francisco Creissac Freitas Campos

Orientador: Ana Cristina Ramada Paiva

---

23 de Julho de 2014

Resumo

Os jogos fazem parte da vida dos humanos desde os tempos mais remotos, estando presentes não só na infância, como na idade adulta. Ao longo dos últimos anos, estudos realizados comprovam que o uso de jogos como um complemento à tradicional aprendizagem é muito mais eficiente do que apenas a utilização do tradicional método de ensino. Os jogos podem ser ferramentas educacionais eficientes, uma vez que enquanto divertem e motivam o utilizador, facilitam a aprendizagem e aumentam a capacidade de retenção do que foi ensinado, exercitando a mente e os pensamentos intelectuais.

Quando os jogos são utilizados para motivar o processo de aprendizagem são definidos como jogos sérios. Portanto, através do uso de mecanismos de jogo, desenvolveu-se o *iLearnTest*, um jogo sério capaz de suportar o ensino de testes de software, que transmite os conceitos exigidos para a obtenção da certificação *Foundation* do ISTQB, permitindo assim a preparação dos alunos para o mundo do trabalho.

De forma envolvente e divertida o utilizador tem a oportunidade de aprender ao seu próprio ritmo, assimilar conceitos teóricos, realizar exercícios práticos e receber uma pontuação pelo seu desempenho.

O *iLearnTest* é uma plataforma *online* com acesso por autenticação, podendo ser acedida simultaneamente por diversos utilizadores. O *iLearnTest* mostra uma tabela com as pontuações dos diferentes jogadores promovendo a competição entre eles e, consequentemente, contribuindo para melhorar a aprendizagem.

Abstract

Games are part of human life since ancient times, being present not only in our childhood but also in other moments. Over the past few years, studies point that the use of games as a supplement to traditional learning is much more efficient than just using the traditional teaching method. Games can be effective educational tools, since as they amuse and motivate, they facilitate learning and increase retention capacity of what was taught, exercising the mind and intellectual thoughts of the player.

When games are used to motivate the learning process they are defined as serious games. So, through the use of game mechanisms, iLearnTest was developed, a serious game capable of withstanding the teaching of software testing, transmitting concepts required to obtain the certification ISTQB Foundation, thus allowing the preparation of students for the working world.

In an engaging and entertaining way the player has the opportunity to learn at his own self-pace, assimilate theoretical concepts, conduct practical exercises and receive a performance score.

iLearnTest is an online platform with access through authentication, that can be accessed simultaneously by several users. iLearnTest shows a table with scores of different players promoting competition among them and thus contributing to enhance learning.

Agradecimentos

Agradeço à minha querida orientadora, Prof. Dra. Ana Paiva, por andar sempre em cima de mim, por toda a disponibilidade e colaboração, por todos os conhecimentos transmitidos e por toda a capacidade de estímulo ao longo do trabalho.

Agradeço aos meus pais, Maria e Manuel, pela educação que me deram, por todo o esforço que fizeram para que eu pudesse estar hoje a terminar este curso, por todo o apoio, por toda a ajuda, por nunca me terem deixado desistir de alcançar este meu sonho e, principalmente, por fazerem tudo por mim. O meu infinito agradecimento.

Agradeço à minha irmã, Márcia, por ser uma verdadeira irmã, por acreditar no meu futuro, por toda a competição saudável que nos faz ser melhor e por toda a ajuda (não só na leitura desta dissertação como em tudo).

Agradeço aos meus amigos de curso, que ao longo destes 5 anos me acompanharam, por toda a ajuda, por todo o incentivo, por todos os inesquecíveis momentos vividos, pelas pessoas espetaculares que são.

E aqueles que aqui não refiro, mas que nunca me esqueço, o meu muito obrigado por acreditarem em mim e estarem sempre lá.

Ninguém vence sozinho, por isso o meu sincero: OBRIGADA A TODOS!

Tânia Ribeiro

"If today were the last day of your life,
would you want to do what you are about to do today?"

Steve Jobs

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

ADDIE	*Analysis Design Development Implementation Evaluation*
ISTQB	*International Software Testing Qualifications Board*
LO	*Learning Objectives*
MADDIE	*Analysis Design Development Implementation Evaluation Management*

Capítulo 1

Introdução

Este capítulo apresenta o equadramento do trabalho, o projeto, a motivação e os objetivos a atingir. Por fim, descreve-se também a estrutura do documento.

1.1 Contexto/Enquadramento

Uma área de investigação em crescimento é o desenvolvimento de jogos para o ensino e treino de conhecimentos em diversas áreas. Um ambiente natural para este tipo de abordagem é o nível inicial de escolaridade onde se pretende motivar e captar a atenção das crianças para a aprendizagem. Contudo, também existem exemplos de desenvolvimento de jogos para adultos em diferentes áreas, como saúde, turismo, gestão de empresas, defesa, etc. Estes são os chamados jogos sérios, nos quais se pretende transmitir um conteúdo de cariz educativo ao utilizador, de forma interativa e entusiasmante [1].

Os jogos sérios são projetados para o ensino ou para a prática dos conhecimentos técnicos e não-técnicos importantes ao desenvolvimento profissional. Porém, incluem elementos lúdicos e de entretenimento para aumentar a atenção e incentivar o envolvimento do utilizador na aprendizagem.

Para ensinar testes de software é necessário abordar uma grande gama de técnicas e conceitos. A realidade enfrentada pelos profissionais desta área não é fácil de imitar numa sala de aula. Os conceitos teóricos podem ter sido assimilados, mas o estudante não percebe como e quando deve aplicá-los no mundo "real". Esta falta de entendimento prático dos alunos, mesmo no final da sua formação acadêmica, não satisfaz os empregadores que têm de contratá-los [2].

O uso de jogos como métodos alternativos de ensino é uma alternativa para a aprendizagem destas matérias. Várias experiências têm vindo a comprovar que desta forma um aluno consegue assimilar realmente o que lhe é ensinado, pondo de lado os seus medos e ansiedades e dedicando-se completamente ao jogo. A concentração é intensa e com isso a matéria é assimilada [3].

1.2 Projeto

Este projeto consiste no desenvolvimento de um jogo sério educacional para o ensino de testes de software com o objetivo de treinar os utilizadores para a obtenção da certificação *Foundation* do ISTQB (*International Software Testing Qualifications Board*) baseado na estrutura de capítulos do programa de certificação *Syllabus* na versão de 2011. Este corpo de conhecimento (*Syllabus*) é estruturado em seis capítulos, apresentados na secção seguinte.

Com o desenvolvimento deste projeto pretende-se abranger diversas características que o tornem atraente para os alunos. Para além de transmitir os conhecimentos atrás mencionados, pretende-se que este jogo educacional:

- Incorpore desafios em cada matéria para promover o envolvimento dos estudantes durante a prática dos conhecimentos e técnicas de teste de software;
- Ofereça a possibilidade de estudo individual para que os alunos possam aprender ao seu próprio ritmo, gastando o tempo que achem necessário para a aprendizagem;
- Atribua uma pontuação em cada jogo concluído pelo estudante, de forma a incentivar a obtenção da pontuação máxima e, por conseguinte, aumentar o seu conhecimento;
- Apresente os resultados dos estudantes após cada desafio, indicando as respostas certas e erradas e apresentando a correção neste último caso.

Aliado a estas características pretende-se que o software esteja disponível através de um *browser* para um acesso simples, em qualquer lugar, e disponível a qualquer um em qualquer altura. Ou seja, é possível o acesso, ao mesmo tempo, por múltiplos jogadores.

1.3 Motivação e Objetivos

Embora os testes de software sejam vistos como uma das principais medidas de qualidade, as técnicas de teste e as boas práticas ainda são raramente aplicadas pelas empresas de desenvolvimento de software. Uma possível razão para esse problema é a falta de profissionais qualificados e disponíveis para implementar essas técnicas e boas práticas [4].

Para superar este problema pretende-se criar uma nova forma de aprendizagem para dar uma visão mais realista do processo de testes de software através do desenvolvimento de um jogo sério. Este jogo permitirá a simulação do processo de uma forma mais agradável e engraçada, em que cada estudante pode treinar ao seu próprio ritmo. Este jogo será relativamente fácil de entender e interessante para atrair os alunos e melhorar a qualidade da aprendizagem.

Através do uso de mecanismos de jogo, pretende-se desenvolver uma ferramenta educativa para suportar o ensino de testes de software de forma envolvente e divertida, em que o utilizador aprende ao seu próprio ritmo, recebe uma avaliação e enfrenta desafios de um verdadeiro jogo.

O ensino de testes de software é, normalmente, incluído nas disciplinas de ensino de engenharia de software o que não permite aos estudantes obter conhecimentos mais pormenorizados sobre

esta matéria. Posto isto, um dos objetivos principais é permitir ao estudante aplicar e treinar os conhecimentos dados nas aulas, de uma forma mais divertida e aliciante.

Através da construção deste jogo sério pretende-se também fornecer conceitos exigidos para a obtenção de uma certificação de nível base do ISTQB, que é a certificação de qualidade internacional atribuída a software *testers*. Para isso serão transmitidos conhecimentos sobre todas as matérias do programa de certificação *SYLLABUS*.

1.4 Estrutura da Dissertação

Para além da introdução, este documento contém mais quatro capítulos. O capítulo 2 descreve o estado da arte e apresentada projetos existentes, e seus problemas, relevantes ao contexto do problema, de forma a demonstrar o que existe na literatura dentro do mesmo domínio. Também faz uma revisão tecnológica às principais ferramentas que poderão ser utilizadas no desenvolvimento do projeto. O capítulo 3 apresenta o *iLearnTest*, os detalhes da sua implementação, os objetivos da aprendizagem que se pretende transmitir, o *website* construído, a descrição da estrutura do jogo, assim como a descrição dos diferentes desafios implementados e os seus objetivos em específico. A descrição da validação realizada ao projeto é feita no capítulo 4. Por fim, o capítulo 5 apresentada as conclusões desta dissertação e o trabalho a realizar no futuro.

Capítulo 2

Revisão Bibliográfica

Neste capítulo faz-se um levantamento de estado da arte relativo a projetos existentes de ensino através de software educativo, os problemas mais relevantes nesta área, e tecnologias e ferramentas existentes de auxílio ao desenvolvimento de software educativo / jogos sérios. O capítulo divide-se nas quatro secções seguintes:

- 2.1 Introdução - faz uma introdução aos jogos sérios;
- 2.2 Metodologias de desenvolvimento de jogos - apresenta as metodologias de desenvolvimento de jogos encontradas na literatura que se mostraram interessantes para a realização do projeto;
- 2.3 Tecnologias de desenvolvimento de jogos - faz uma revisão das tecnologias se suporte ao desenvolvimento de jogos e das suas características;
- 2.4 Exemplos da literatura - descreve exemplos da literatura encontrados, tanto no domínio da engenharia de software como no domínio específico dos testes de software;
- 2.5 Resumo e Conclusões - descreve as conclusões retiradas do estudo realizado.

2.1 Introdução

Para motivar e envolver os alunos e, consequentemente, melhorar a qualidade da aprendizagem, alguns investigadores sugerem novas formas de ensino, incluindo o uso de jogos sérios nas salas de aula [2]. Estudos anteriores, também, revelam que o uso de jogos como um complemento à tradicional aprendizagem é muito mais eficiente do que apenas a utilização do tradicional método de ensino [5, 6, 7, 8, 9, 10].

Um jogo sério é uma experiência concebida, usando mecanismos e raciocínios de jogo, para educar/ensinar indivíduos num domínio de conteúdo específico. Há jogos sérios para o ensino de liderança, de técnicas de venda, e de outros tópicos de negócios, assim como no campo da saúde.

Os jogos sérios são vistos como um uso nobre de mecanismos de jogo e uma forma de envolver e interagir com os alunos [1].

Com o presente estado da arte pretende-se fazer um estudo das metodologias e tecnologias de desenvolvimento de jogos que poderão ser benéficas para a criação deste jogo. Assim como dos jogos já existentes no domínio de engenharia de software e de testes de software, de forma a perceber o que já foi feito e com que aspetos se poderá inovar.

2.2 Metodologias de Desenvolvimento de Jogos

Na literatura existem diversos trabalhos que descrevem metodologias de desenvolvimento de jogos, e de jogos sérios, ou ainda processos e artefactos que fazem parte do ciclo de vida de criação de jogos, tais como modelos conceituais [11]. Nesta secção são apresentadas metodologias de desenvolvimento de jogos que se demonstraram relevantes para o projeto a desenvolver.

A secção 2.2.1 descreve as metodologias de desenvolvimento ADDIE e MADDIE, que normalmente se encontram associadas à criação de ferramentas de ensino supervisionadas por um professor. O ponto diferenciador destes dois processos é a inclusão de técnicas de gestão de projetos na metodologia MADDIE.

O ponto 2.2.2, Método Ágil SCRUM, pormenoriza o processo ágil SCRUM de desenvolvimento de software e, por consequência, também de jogos sérios. Este utiliza uma abordagem iterativa ideal para projetos complexos, imprevisíveis e de larga escala.

A secção 2.2.3 descreve a metodologia Hybrid proposta pelo livro [1], que tira partido dos pontos fortes das metodologias mencionadas nos dois pontos anteriores. Uma fusão desenvolvida e testada pelo autor, capaz de se adaptar a cada projeto de acordo com o tamanho, o âmbito e o nível de complexidade.

A secção 2.2.4 traça a estrutura básica para o processo geral de produção de jogos segundo a perspetiva do produtor, apresentada no livro "Manual de Produção de Jogo Digitais" [12] por Chandler.

Por fim, a secção 2.2.5 expõem a metodologia proposta por Schuytema, no livro "Design de Games: Uma Abordagem Prática" [13]. Esta é uma abordagem para o desenvolvimento de jogos constituída por três grandes períodos no ciclo de desenvolvimento, pelo ponto de vista do designer: pré-produção, produção e pós-produção.

2.2.1 Metodologia ADDIE

A metodologia de desenvolvimento ADDIE [1] é um processo de modelo para a criação de software, normalmente associado à criação de ferramentas de ensino supervisionadas por um professor. Esta é usada por muitas empresas de *design* e de desenvolvimento de *e-learning*.

Este método envolve uma abordagem linear em cascata, constituída por cinco etapas individuais e semi-discretas: análise, *design*, desenvolvimento, implementação e avaliação (figura 2.1). Ocasionalmente, este processo é referido como MADDIE, quando é incluída a etapa de gestão de projeto.

Na fase da análise, é analisado o tipo de problema a ser resolvido com a aprendizagem, e o tipo de conteúdo a ensinar, determinando, por exemplo, se é conhecimento declarativo ou de resolução de problemas. Nesta fase é necessário identificar também o tipo de aprendizes, definindo o seu conhecimento atual e a sua predisposição para a aprendizagem de novos conhecimentos, assim como as tecnologias disponíveis para o desenvolvimento da ferramenta de aprendizagem.

Uma vez que a análise se encontre completa, o passo seguinte é o *design* da solução. O resultado desta fase é um documento de *design* onde estão especificados os objetivos da aprendizagem e a estratégia de ensino a adotar, assim como os tópicos de avaliação sumativa. Normalmente este documento é assinado pelo cliente de forma a fechar o acordo para poder ser iniciada a fase de desenvolvimento sem o risco de mudanças inesperadas.

A programação e a criação da aprendizagem acontece na etapa de desenvolvimento. Nesta etapa são criadas as interfaces necessárias para a ferramenta e é realizada uma avaliação formativa com aprendizes e instrutores. No caso de ser necessário realizar modificações estas podem ser incorporadas diretamente no desenvolvimento, ou o processo pode voltar à fase de *design*.

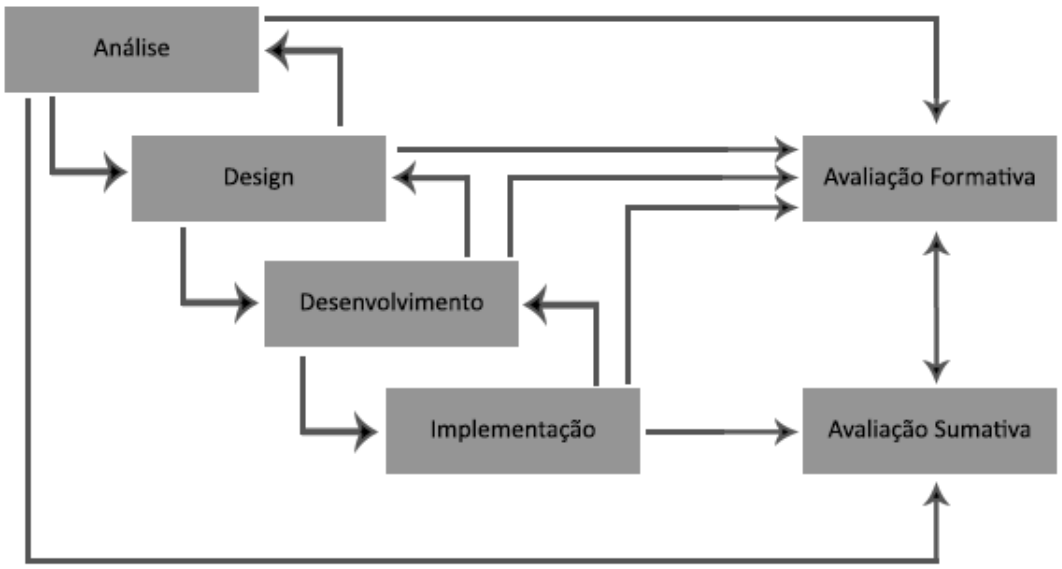


Figura 2.1: Modelo ADDIE.

Posteriormente, a implementação é executada. Nesta fase é realizada a demonstração da ferramenta aos utilizadores, tanto alunos como professores, avaliando se compreendem a sua utilização. Nesta etapa é importante perceber se a ferramenta se encontra funcional em todos os dispositivos exigidos.

A fase de avaliação ocorre não só no fim mas também durante todo o processo e divide-se em dois tipos. A avaliação formativa acontece durante as fases de *design* e de desenvolvimento, onde os materiais são examinados e são realizadas alterações sempre que necessário. Já a avaliação sumativa ocorre no final do processo, onde é avaliada a utilidade da ferramenta de aprendizagem construída.

A metodologia MADDIE difere da ADDIE no ponto em que existe gestão de projeto através de um gestor de projeto, que lidera a equipa, constituída por designers, artistas gráficos e programadores [1].

2.2.2 Método Ágil SCRUM

A metodologia de desenvolvimento SCRUM é um método ágil que utiliza uma abordagem iterativa ideal para projetos complexos e imprevisíveis. Esta é normalmente associada a projetos de desenvolvimento de software de larga escala, sendo utlizada por empresas de criação de jogos *online* para múltiplos jogadores de larga escala para atualizar e manter os seus produtos.

Enquanto as metodologias de desenvolvimento tradicionais assentam num desenvolvimento sequencial e segmentado esta assenta no desenvolvimento do produto como um todo, realizado por uma equipa multidisciplinar que trabalha durante todo o processo.

Este método ágil permite à equipa de desenvolvimento escolher a quantidade de trabalho a realizar e decidir qual a melhor forma para o fazer, criando, assim, um ambiente de trabalho flexível e mais produtivo.

Este processo inicia-se quando o cliente propõem a sua ideia ou visão do produto à equipa de desenvolvimento. A partir desta ideia ou visão do produto é criada uma lista de requisitos, o chamado *product backlog*. Esta lista é incompleta devido a novos requisitos que irão surgir durante o ciclo de vida do projeto e que serão incorporados numa das iterações da implementação, às quais se dá o nome de *sprints*. O cliente é responsável por atribuir prioridades aos requisitos do *product backlog*, baseado nos pontos de esforço atribuídos pela equipa. Estas prioridades podem ser alteradas durante o ciclo de vida do projeto.

Para dar início ao projeto, a equipa reúne-se com o cliente para escolher quais os requisitos para o *sprint backlog*, e trabalha a partir deste ponto apenas nesses requisitos. A duração dos *sprints* pode variar entre duas a cinco semanas, ou mais, mas devendo sempre ser mantida curta, com um dia no início para planear o *sprint* e um no fim para ser feita uma revisão do trabalho realizado. Durante o *sprint* não devem ser feitas alterações ao *sprint backlog*.

Este método permite que o software seja desenvolvido rápido e que a inserção de alterações no processo seja fácil, o que melhora a qualidade do produto final e reduz o risco de insucesso.

Para manter a equipa focada no produto final são realizadas reuniões diárias, às quais se dá o nome de *scrums*. Estas reuniões agrupam a equipa completa e têm como objetivo a resposta a três perguntas por parte de todos os elementos. As perguntas são:

- O que é que fiz desde a última reunião?
- O que é que ainda preciso de fazer?
- Quais foram os obstáculos que encontrei?

Para que estas reuniões se realizem de forma correta e garantir que todos os elementos da equipa se encontram na mesma direção e que o projeto é revisto como um todo é elegido um

mestre de SCRUM. Este não é um líder de equipa, é simplesmente responsável por todo o processo de SCRUM: ensinar a todos os intervenientes do projeto as regras deste método; implementar o método para que ele se encaixe na cultura da organização e forneça os resultados esperados; e assegura que todos seguem as regras e práticas de SCRUM.

Todo o processo do projeto é rastreado com um gráfico, o qual de denomina por *burn down chart*, onde é possível visualizar quanto trabalho falta fazer em relação ao tempo que ainda resta. Normalmente este gráfico é colocado num local onde toda a equipa pode observar facilmente, existindo um para cada *sprint* e um para o processo completo do projeto. A figura 2.2 apresenta um exemplo de um *burn down chart* para o processo total de um projeto, onde é possível visualizar os pontos totais realizados em cada *sprint* e os pontos planeados para a realização [1].

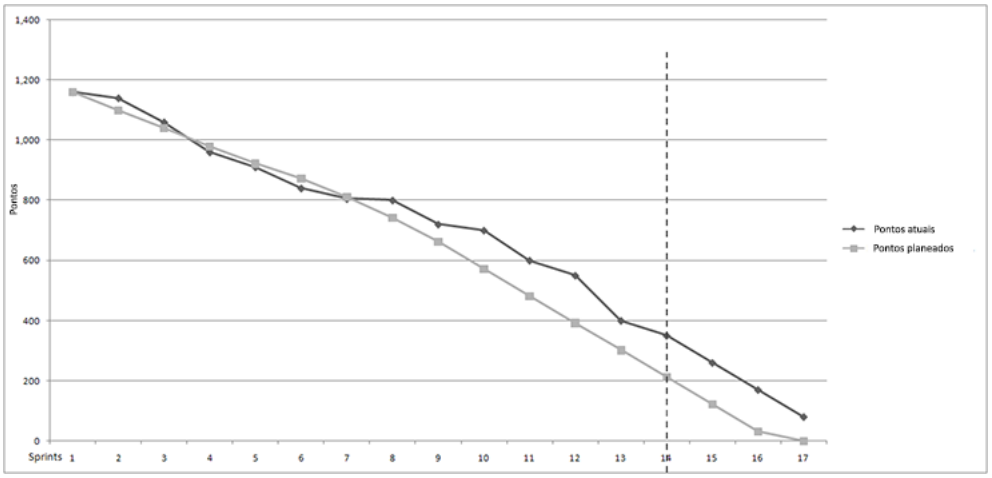


Figura 2.2: *Burn down chart* de exemplo.

A tabela 2.1 apresentada de seguida expõem uma breve descrição dos artefactos que constituem a metodologia SCRUM:

2.2.3 Metodologia Hybrid

A maior parte do esforço realizado pelas empresas de desenvolvimento de jogos sérios não envolve a quantidade de trabalho requerida em projetos de larga escala, portanto o método SCRUM é demasiado para projetos de pequena escala. Mas, por outro lado, a metodologia ADDIE não permite a fácil integração de alterações durante o processo de desenvolvimento. Posto isto, o livro [1] propõem uma nova metodologia criada a partir da fusão destes dois métodos, a metodologia Hybird. Esta é capaz de se adaptar de acordo com o tamanho, o âmbito e o nível de complexidade de cada projeto.

No primeiro passo, equivalente à fase de análise do modelo ADDIE, pretende-se determinar quais os conhecimentos a ensinar, o tipo de conteúdo e os domínios, assim como os objetivos de aprendizagem relativos a cada tipo de conteúdo. Neste ponto é importante determinar se a motivação dos aprendizes é intrínseca ou extrínseca, uma vez que terá impacto na criação das recompensas e do sistema de pontos.

Tabela 2.1: Tabela de artefactos do método SCRUM.

Artefacto	Descrição
Product Backlog	Lista de requisitos que o projeto deve conter, priorizados pelo cliente.
Sprint Backlog	Lista de tarefas a serem realizadas em cada iteração (*sprint*) do projeto. Esta lista é definida pelo cliente e pela equipa de desenvolvimento no início de cada iteração.
Sprint	Ciclo de trabalho no método SCRUM. Esse ciclo pode ter a duração de duas semanas a cinco semanas, mas todos as iterações devem ter sempre a mesma duração. Em cada *sprint* devem ser implementadas as tarefas definidas pelo *sprint backlog*.
Burn Donw Chart	Gráfico onde é possível visualizar quanto trabalho falta fazer em relação ao tempo restante. Normalmente existe um para cada sprint e um para o processo completo do projeto.

Na segunda etapa é reunido um grupo de entendidos na matéria (programadores, artistas e *designers* de jogos) para a realização de um *brainstrom* para a captação de ideias. Daqui resulta a quantidade de conteúdo certa e de conhecimento do processo necessária para a realização do projeto, juntamente com o protótipo, os momentos de ensino no jogo e as tecnologias a utilizar para a sua implementação. Este passo incluí o desenvolvimento base da história do jogo, do sistema de pontos e recompensas e das atividades de aprendizagem. No final é elaborado o documento de *design* com o conteúdo resultante do *brainstrom*.

No primeiro *sprint* é importante experimentar a jogabilidade com o protótipo desenvolvido, de forma a avaliar o nível de diversão que o jogo transmite e, caso necessário, serem realizadas alterações. Em paralelo a este, deverá ser criado outro *sprint*, com um artista e um *designer* de jogos de ensino para criarem a história de jogo e a arte conceitual. O objetivo é colocar em papel toda a sequência de atividades do jogo.

Assim que a sequência de jogo esteja criada deve ser testada com utilizadores reais da ferramenta. Estes podem realizar comentários e responder a perguntas providenciadas pela equipa de desenvolvimento. Logo que os comentários sejam obtidos devem ser incorporados no documento de *design* e no protótipo.

A partir deste ponto é iniciado o *sprint* de desenvolvimento, começando por criar o primeiro nível de funcionalidade de jogo. Depois é realizada uma reunião com o cliente para demonstrar os resultados e realizar alterações se necessário, e é dado início a um novo *sprint*.

As reuniões diárias de equipa, realizadas na metodologia SCRUM, são também importantes durante o processo de desenvolvimento neste método. Depois de cada *sprint* é realizado um teste de jogo às novas funcionalidades na perspetiva de jogo e de ensino.

Quando o projeto está finalmente completo é realizada uma avaliação sumativa para avaliar a utilidade da ferramenta de ensino desenvolvida, assim como acontece no método ADDIE [1].

2.2.4 Metodologia de Chandler

No livro "Manual de Produção de Jogo Digitais" [12] Chandler descreve uma estrutura básica para o processo geral de produção de jogos, sob a perspetiva do produtor.

A fase inicial é a pré-produção, onde é definido o conceito do jogo e os seus requisitos quanto à arte, ao design e à mecânica de jogo. Nesta fase é também feito o planeamento do projeto e uma avaliação de risco.

Após concluída a fase inicial, é realizada a produção. Nesta fase é realizada a implementação do plano estruturado, assim como o rastreamento do projeto. A avaliação de risco é novamente realizada, de forma a prever possíveis problemas futuros para que possam ser corrigidos atempadamente.

De seguida é executada a etapa de teste, são feitos os testes para a validação do plano e o código é libertado para experiências, dando-se então início à fase de pós-produção. Nesta fase final são realizados os testes de aprendizagem com aprendizes reais e é desenvolvido o plano de arquivamento.

Em jogos maiores e com riscos altos, o processo de produção é iterativo e com vários ciclos de produção, por exemplo, execução das quatro etapas para a criação do primeiro protótipo jogável, seguido da execução novamente de todas as etapas para produção do segundo protótipo jogável, e assim por diante até à versão final [11].

2.2.5 Metodologia de Schuytema

No livro "Design de Games: Uma Abordagem Prática" [13] é descrita pelo autor, Schuytema, uma abordagem para o desenvolvimento de jogos constituída por três grandes períodos no ciclo de desenvolvimento, pelo ponto de vista do designer: pré-produção, produção e pós-produção.

O passo inicial, designado de pré-produção, serve para a conceção do documento de *design*, em que através de um *brainstorming* entre os intervenientes do projeto é concebido o conceito do jogo e todos os seus requisitos.

Na segunda etapa realiza-se a construção do jogo e a programação do código fonte da aplicação segundo os requisitos definidos na fase inicial.

Por fim, na pós-produção é executada a inclusão de recursos e feita a avaliação da recetividade por parte dos utilizadores, analisando se estes compreendem a utilização da ferramenta.

Sendo esta uma metodologia para *designers* tem a desvantagem de não contemplar os processos de modelagem e simulação [11].

2.3 Tecnologias para o Desenvolvimento de Jogos

Nos dias de hoje, o aparecimento de novas tecnologias para o desenvolvimento de jogos é cada vez maior, por isso torna-se importante o estudo detalhado das características que oferecem para o desenvolvimento de um bom projeto. Nesta secção são, então, descritas as tecnologias estudadas, para o desenvolvimento de jogos, que demonstraram as melhores características para o desenvolvimento deste projeto.

O ponto 2.3.1 apresenta a plataforma de criação de jogos 2D para computadores, dispositivos móveis e *web*: Stencyl. O segundo ponto, 2.3.2, descreve o Construct 2: software de desenvolvimento de jogos 2D em *5*, destinado principalmente a utilizadores sem conhecimentos de programação, uma vez que apresenta um sistema de *drag-and-drop*, um editor visual e um sistema de lógica baseada em comportamentos. A secção 2.3.3 apresenta o Unity: uma ferramenta de criação de jogos 3D e 2D, usada para desenvolver jogos para *web*, plataformas *desktop*, consolas e dispositivos móveis. E por fim, a secção 2.3.4 descreve a ferramenta GameSalad que utiliza o sistema de *drag-and-drop* de forma a facilitar o desenvolvimento de jogos a utilizadores sem conhecimentos de programação.

2.3.1 Stencyl

Stencyl [14] é uma plataforma de criação de jogos 2D para computadores, dispositivos móveis e *web*. Esta está disponível para MAC, Windows e Linux, e em duas versões, uma gratuita, e uma disponível para compra que oferece uma coleção maior de características.

Esta tecnologia oferece a possibilidade de criação de jogos de uma forma intuitiva e aceleradora do fluxo de trabalho, assim como a publicação dos jogos em todas as principais plataformas (iOS, Android, Flash, Windows, Linux e Mac) sem a necessidade de escrever código. Importante salientar que a versão grátis desta ferramenta permite a publicação de jogos na web (Flash), mas apenas o teste em dispositivos iOS e Android, não permite a publicação para estas plataformas.

O sistema de *drag-and-drop* oferece uma forma simples de desenvolvimento de jogos para aqueles que não têm conhecimentos técnicos de programação. Por outro lado, os utilizadores avançados podem usar o código para adicionar novos recursos e criar comportamentos complexos.

O Stencyl incluí vários módulos para o desenvolvimento dos jogos: um para criar e editar o código e a lógica do jogo em peças modulares, conhecidos como comportamentos e eventos; outro para importar e editar *tilesets*, incluindo formas de colisão, aparência e animações; outro para criar e editar atores de jogo e as suas definições, incluindo comportamentos, física e animações; e por fim, um editor para criar e editar os níveis e os estados do jogo.

As ferramentas adicionais permitem que o utilizador importe imagens para usar como primeiro plano e plano de fundo nas cenas, fontes de importação e edição, sons de importação e arquivos de música e alterar as configurações do jogo (como, por exemplo, teclas de comando do jogador e a resolução do jogo).

A necessidade de criação de novos comportamentos de jogo é reduzida através da biblioteca de comportamentos que está incluída no Stencyl. Alternativamente, esta ferramenta pode ser

configurada para utilizar editores de imagem externos, como Photoshop e GIMP, para modificar as imagens que já estão carregadas num projeto.

2.3.2 Construct 2

Construct 2 [15] é um software de desenvolvimento de jogos 2D em *HTML5*, destinado principalmente a utilizadores sem conhecimentos de programação, uma vez que apresenta um sistema de *drag-and-drop*, um editor visual e um sistema de lógica baseada em comportamento. Para programadores, a ferramenta apresenta um editor de *JavaScript* permitindo assim a criação de comportamentos mais complexos.

Este sistema oferece comportamentos flexíveis, previsualização instantânea, efeitos visuais aliciantes, publicação em todas as plataformas e fácil extensibilidade através da agregação de diversos *plugins*.

Assim como a tecnologia apresentada no ponto anterior esta, também, apresenta uma versão grátis e uma paga. A versão grátis para além de permitir a publicação *web* (em *HTML5*), também permite a publicação no mercado do Windows 8, na loja *online* do *browser* Chrome, no centro de jogos do Facebook e na loja dos criadores de Construct 2. Para a publicação de jogos em iOS, Android e Linux é necessário comprar a versão paga. A versão grátis está limitada ao uso de cem eventos, quatro camadas e dois efeitos.

2.3.3 Unity

O Unity [16] é uma ferramenta de criação de jogos 3D e, mais recentemente, também de jogos 2D. Esta é usada para desenvolver jogos para *web*, plataformas *desktop*, consolas e dispositivos móveis. Inicialmente desenvolvida para MAC OS X, esta plataforma cresceu desde 2005 até aos dias de hoje para um motor de jogo multiplataforma.

Atualmente, o Unity suporta desenvolvimento para iOS, Android, Windows, BlackBerry 10, MAC OS X, Linux, web, Flash, PlayStation 3, PlayStation Vita, Xbox 360, Windows Phone 8 e Wii U. Este sistema possuí duas versões: a versão paga (Unity Pro) e a versão gratuita (Unity) que pode ser usada tanto para fins educacionais, como para fins comerciais. Além disso, a versão Unity Pro está disponível para download na página online [16] podendo ser testada por um período de 30 dias. As desvantagens da versão gratuita em relação à versão grátis restringem-se apenas em alguns efeitos extra que podem ser aplicados no jogo.

A partir do momento em que utilizador abre o editor do Unity é presenteado com fluxos de trabalho intuitivos apoiados por um poderoso conjunto de ferramentas, para diminuir o tempo de desenvolvimento do jogo. O utilizador pode importar instantaneamente qualquer tipo de conteúdos da loja e criar mundos complexos, com blocos de construção.

Uma vantagem experimental vital para o trabalho do utilizador são os testes reprodução feitos em simultâneo com a edição de conteúdos, tudo numa só janela.

2.3.4 GameSalad

GameSalad [17] é uma ferramenta que utiliza o sistema de *drag-and-drop* de forma a facilitar o desenvolvimento de jogos a utilizadores sem conhecimentos de programação. Com editores visuais e uma lógica baseada em comportamentos, esta é usada por utilizadores e profissionais criativos, como *designers* gráficos, artistas e desenvolvedores de jogos, para uma rápida prototipagem e construção de jogos multiplataforma de auto-publicação.

Este sistema fornece uma interface gráfica para descrever as regras e os comportamentos dos objetos do jogo, sem o utilizador sentir a necessidade de saber programar. Os comportamentos são componentes de um ator que pode instantaneamente ou persistentemente afetar o ator dependendo das regras que o compõem. A aplicação vem com uma biblioteca de comportamentos (movimentos, estados de atributos, colisão, etc.) que podem ser agrupados para criar novos comportamentos. As características principais que esta ferramenta apresenta são:

- Publicação em múltiplas plataformas – como iOS, MAC OS X, Android, Windows 8, e para a *web* em *HTML5*;
- Edição em tempo real – os utilizadores podem editar a cena de jogo enquanto este está a ser executado;
- Pré-visualização do jogo – contém um modo de visualização específico para testar os jogos nos dispositivos vendo o seu desempenho e gasto de memória;
- Editor de cenas – os utilizadores podem adicionar e manipular os atores numa cena através do sistema simples de *drag-and-drop*;
- Expressões - para utilizadores avançados, o GameSalad contém um editor para definir comportamentos complexos com expressões matemáticas e uma biblioteca de funções.

Assim como todas as tecnologias apresentadas anteriormente esta também está disponível em duas versões. A versão gratuita contém todas as funcionalidades da versão paga exceto a publicação em Windows 8 e Android.

2.4 Exemplos da Literatura

Na criação de um novo jogo educacional é importante o estudo sobre jogos já existentes no mesmo domínio, de modo a perceber aquilo que já foi feito e qual poderá ser o ponto diferenciador do novo jogo. Esta secção apresenta exemplos de jogos sérios existentes na literatura relevantes para a realização do projeto, no domínio da engenharia de software e no domínio específico de testes de software.

2.4.1 Jogos Sérios para o Ensino de Engenharia de Software

Os jogos sérios permitem que os jogadores aprendam e pratiquem conhecimentos técnicos e não técnicos que são importantes para melhorar o seu desenvolvimento profissional. No domínio do ensino de engenharia de software existem inúmeros jogos desenvolvidos, esta secção apresenta quatro deles que se demonstraram mais interessantes tanto no ponto de vista lúdico como da aprendizagem.

2.4.1.1 SimSE

O SimSE, segundo [9], é um jogo educacional que decorre num ambiente de simulação, cujo objetivo é aumentar os conhecimentos de engenharia de software aproximando-se da realidade praticada nas empresas de desenvolvimento de projetos.

Este jogo é destinado a um único jogador que deve assumir o papel de gestor de projeto de uma equipa de programadores. O jogador pode praticar num ambiente virtual de engenharia de software de forma gráfica, interativa e divertida num cenário em que recebe uma avaliação direta do seu desempenho.

Ao gerir o projeto, o jogador tem como funções contratar e despedir funcionários, atribuir tarefas aos seis engenheiros de acordo com os seus pontos fortes, monitorizar o desempenho da equipa, adquirir ferramentas que levem à melhoria do projeto, entre outras funções. O jogador pode ainda motivar os engenheiros de software com aumento de salário ou bónus após a entrega do projeto.

Caso o jogador não entregue dentro do prazo e/ou o custo do projeto exceder o limite, perde o jogo; caso contrário, recebe uma pontuação ao entregar o software com base no seu desempenho.

Um ponto fulcral neste jogo é a interface gráfica como é visível na figura 2.3. Nela é apresentado um escritório virtual muito aproximado da realidade, onde existem cadeiras, mesas, computadores e salas de reuniões. Na janela de jogo o jogador tem acesso a toda a informação importante para a gestão do projeto, como informação sobre os seus funcionários (a produtividade de cada um, o nível de energia e a tarefa que está a realizar), o estado de desenvolvimento das tarefas (o tamanho, a integridade da tarefa e o nível de correção), o nível de satisfação dos clientes, o orçamento e o prazo para a conclusão do projeto, assim como o rendimento conseguido com as ferramentas utilizadas.

No mundo real, as metodologias de desenvolvimento de software variam de acordo com as empresas e, como tal, o SimSE é capaz de retratar as diferenças que possam existir nesses processos, uma vez que permite a personalização dos seus modelos de processo.

2.4.1.2 SE•RPG

O jogo SE•RPG [18] simula um ambiente de desenvolvimento de software utilizando um cenário de uma empresa de desenvolvimento fictícia com diversas personagens, com as quais o jogador interage, como é possível observar na figura 2.4.

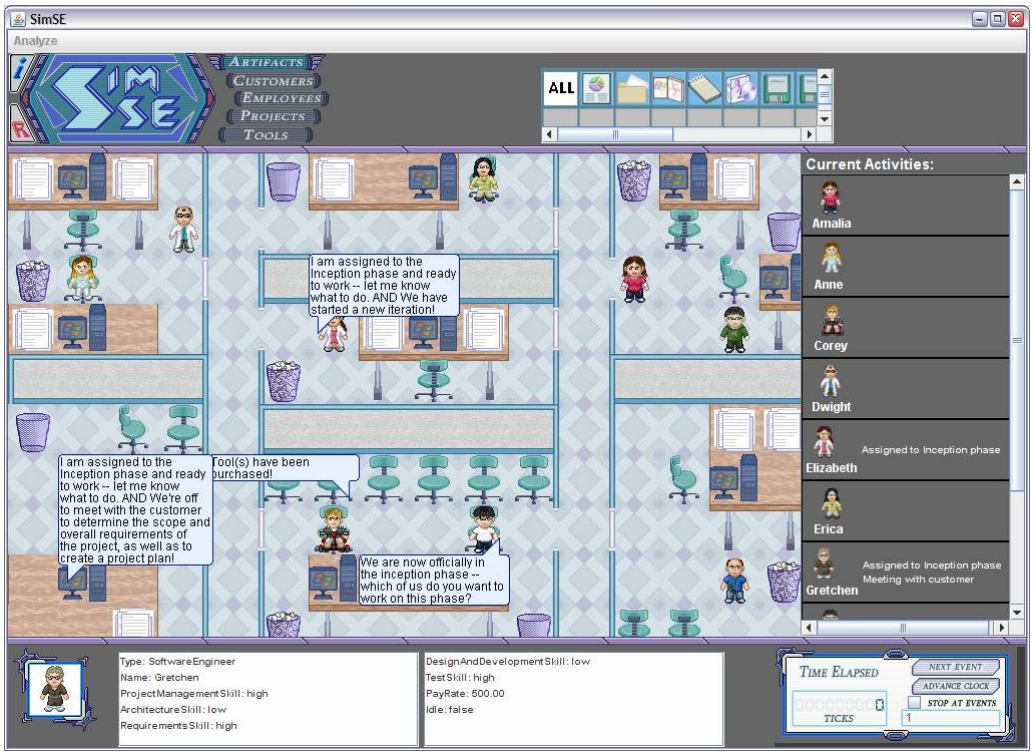


Figura 2.3: Interface do jogo SimSE [9].

No início do jogo, é apresentada breve descrição do projeto selecionado pelo jogador contendo o orçamento e o prazo para o seu desenvolvimento. Através desta descrição, o jogador deve definir qual será a metodologia de desenvolvimento que pretende utilizar das duas opções disponíveis, modelo cascata e modelo iterativo, assim como a linguagem de programação a que será utilizada.

De seguida, cabe ao jogador escolher a equipa de programadores para o desenvolvimento do projeto com base nas habilidades apresentadas por cada personagem. O jogador dá início ao desenvolvimento do software atribuindo tarefas a todos os elementos da equipa através da lista de requisitos disponível. A partir deste ponto o jogador pode controlar o desenvolvimento do projeto, verificando o tempo gasto e o orçamento restante.

Durante o desenrolar do jogo, o jogador pode contratar ou despedir funcionários, sendo possível verificar o progresso de cada etapa e a produtividade de cada membro da equipa. No final da implementação o projeto é concluído, o software é entregue ao cliente e o jogador recebe a sua pontuação (com base nas metas alcançadas, segundo o prazo de entrega e o custo total do projeto).

2.4.1.3 PlayScrum

O PlayScrum apresentado em [2], representa uma primeira tentativa de utilizar um jogo de cartas para ensinar o método ágil SCRUM. Este aborda muitas das fraquezas das abordagens mais tradicionais de aprendizagem e traz benefícios adicionais na forma de aprendizagem. Permite aos alunos adquirir uma sólida compreensão das técnicas de engenharia de software do mundo real que podem ter sido mal compreendidas ou ignoradas completamente durante o ensino escolar.

Figura 2.4: Interface gráfica do jogo SE•RPG [18].

Este jogo apresenta uma natureza muito visual, é simples e divertido de jogar, permite a aprendizagem colaborativa e fornece uma avaliação quase imediata aos jogadores sobre os conhecimentos aprendidos.

PlayScrum é um jogo de competição entre dois a cinco jogadores em que todos gerem o mesmo projeto desemprenhando o papel de mestre de SCRUM, ganha o jogador que conseguir concluir primeiro o software. Para ganhar os jogadores devem ser capazes de terminar o projeto dentro do orçamento, satisfazendo as exigências do cliente e produzindo um software de elevada qualidade.

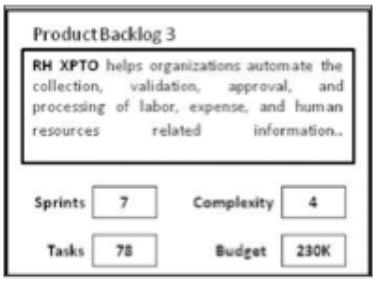


Figura 2.5: Cartão do *product backlog* do jogo PlayScrum [2].

O jogo é dividido em iterações que diferem de projeto para projeto e durante as quais cada jogador deverá desenvolver um número de tarefas definido no início do jogo. Antes do início do jogo é escolhido um cartão com o *product backlog* (figura 2.5), que define o número de *sprints* do projeto e o número de tarefas que devem ser realizadas para terminar o jogo, assim como a complexidade do projeto e o orçamento disponível para cada jogador. Cada jogador recebe duas cartas de programadores (figura 2.6), com as respetivas características, que representam a sua equipa.

De seguida, os jogadores montam o tabuleiro e separam as cartas em quatro montes: um para as cartas de programadores, outro para conceitos e problemas, outro para artefactos azuis, e um para artefactos vermelhos. Em cada jogada, o jogador lança o dado e joga de acordo com o número obtido:

- Se o número for 1, 2 ou 3 o jogador apenas pode tirar cartas do monte de problemas e conceitos;
- Se o número for 4, 5 ou 6 o jogador deve retirar três cartas do monte de problemas e conceitos e retirar do monte de cartas de programadores o valor do número obtido no dado menos três.

O jogador pode guardar na sua mão as cartas retiradas ou colocar imediatamente no tabuleiro, podendo ter apenas um máximo de seis cartas na sua mão, caso contrário as cartas em excesso devem ser descartadas.

Figura 2.6: Exemplo de carta de programador do jogo PlayScrum [2].

Durante a jogada o jogador pode contratar ou despedir programadores, inspecionar os seus artefactos para verificar a existência de erros (esta ação tem o custo de um ponto), ou ainda corrigir os erros dos seus artefactos após a inspeção (esta ação tem também o custo de um ponto).

O *product backlog* demonstrado a cima é dividido em iterações de quatro rodadas em que cada jogador deve tentar adquirir o número máximo de artefactos sem erros. No final de cada iteração o mestre de SCRUM deve apontar os pontos de cada jogador de acordo com o número de artefactos de cada um. Após essa verificação, o mestre descarta os artefactos de cada jogador e o processo repete-se pelas seguintes iterações.

O vencedor do jogo é o jogador que atingir, numa das iterações do projeto, o número de tarefas sem erros pretendidas pelo *product backlog*. Em caso de empate verifica-se qual o jogador que tem menos tarefas com erros.

2.4.1.4 MO-SEProcess

O MO-SEProcess [19] é um jogo de processos de engenharia de software *online* para múltiplos jogadores com base no jogo SimSE, apresentado anteriormente. Este foi concebido para simular

a gestão de um projeto de desenvolvimento de software, desenvolvido no famoso jogo *online* "Second Life".

Este jogo educacional apresenta um ambiente gráfico 3D (representado na figura 2.7) muito próximo da realidade, onde os jogadores podem interagir com outros jogadores e com os objetos presentes no cenário.

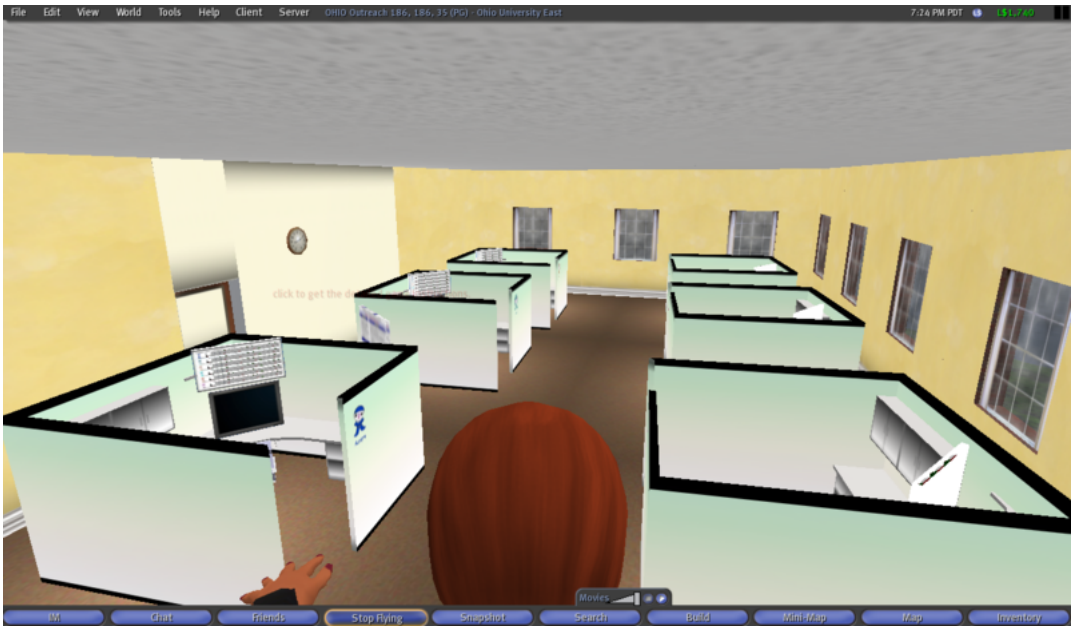


Figura 2.7: Interface gráfica do jogo *MO-SEProcess* [19].

Os jogadores colaboram em equipa para desenvolver um sistema de software. No jogo são fornecidos seis tipos de papéis (por exemplo, gestor de projeto, programador ou *designer*) dos quais cada jogador deve escolher um para desempenhar durante o desenrolar do jogo. Cada papel tem as suas características e tem associado tarefas que devem ser resolvidas durante o jogo.

Durante o jogo, o jogador pode interagir com outros jogadores utilizando vários meios de comunicação já implementados no jogo "Second Life". A pontuação da equipa é dada no final do jogo, considerando o tempo de entrega do produto, o trabalho realizado e a colaboração entre os jogadores.

2.4.2 Jogos Sérios para o Ensino de Técnicas de Teste de Software

Na área de ensino de técnicas de teste de software foram apenas encontrados, na literatura, dois jogos (U-TEST e TestEG) e um tutorial *online* (Bug Hunt). Esta secção descreve as características de cada um deles e as regras de jogo, assim como os conceitos que abrangem dentro da área.

2.4.2.1 U-TEST

O U-TEST [4] é um jogo educacional desenvolvido para dar apoio ao ensino de testes de software. Este jogo tem um desenvolvimento baseado na metodologia ADDIE apresentada anteriormente neste documento.

Neste jogo é esperado que os jogadores tenham um conhecimento básico de conceitos de programação, de engenharia de software e de testes de software. Os seus objetivos baseiam-se no reconhecimento e compreensão dos conceitos principais de testes de software e na perceção e aplicação das técnicas de seleção de entrada de dados, partição da classe de equivalência e análise do valor limite.

Este jogo de simulação foca-se em testes unitários e nas técnicas de caixa-preta, abordando de uma perspetiva teórica e prática, onde o jogador é visto como um candidato a um lugar numa empresa de software. O jogo apresenta uma pequena descrição sobre a empresa e o projeto de que o jogador fará parte, de seguida o jogador deve construir casos de teste para as funções que lhe são apresentadas. Na figura 2.8 é possível ver o ecrã do primeiro desafio que é apresentado ao jogador.

Durante o jogo, o jogador é presenteado com dez etapas, em que cinco delas contém os seguintes desafios:

- Identificar as classes de equivalência;
- Definir valores-limite para as classes identificadas;
- Selecionar o valor correspondente ao valor identificado;
- Configurar um gráfico de causa-efeito ou de árvore de decisão;

No final de cada etapa o jogador recebe uma avaliação do seu desemprenho através de um gráfico e de comentários. No final do jogo o jogador é informado sobre o seu desempenho no geral e da sua posição na tabela de pontuações dos jogadores.

2.4.2.2 Bug Hunt

O tutorial *online* Bug Hunt [20] foi desenvolvido para cativar e aumentar o nível de interesse dos alunos na aprendizagem de técnicas de teste de software. Este contém diversas características que aliciam tanto alunos como professores:

- Incorpora desafios em cada aula e providência uma avaliação imediata do desempenho para aumentar o interesse do aluno enquanto pratica conhecimentos fundamentais de testes de software;
- O aluno pode resolver os desafios ao seu próprio ritmo, despendendo o tempo que achar necessário para a aprendizagem dos conceitos;
- A solução que apresenta é configurável segundo os requisitos de cada professor;

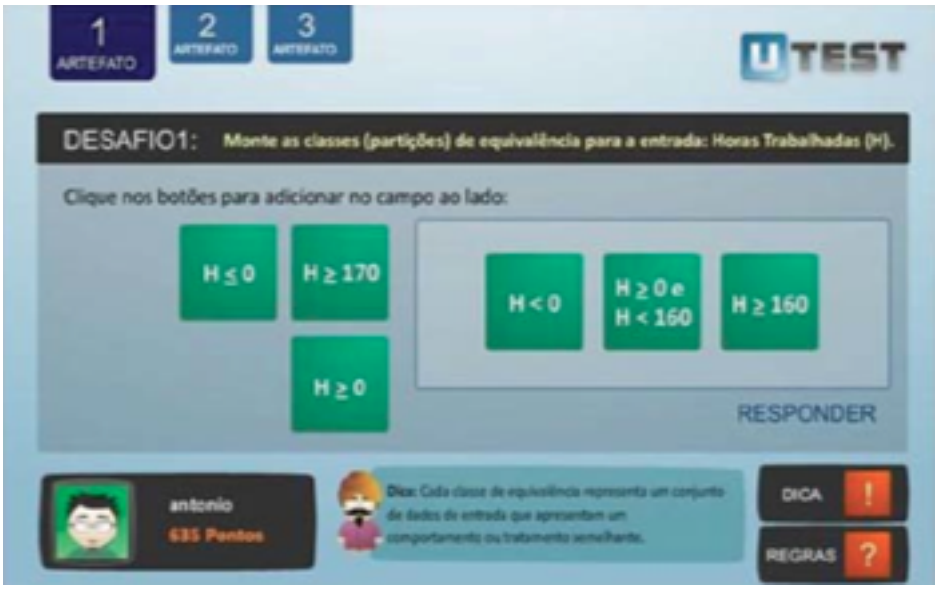


Figura 2.8: Ecrã do primeiro desafio do jogo U-TEST [4].

- O professor recebe a avaliação do desempenho dos seus estudantes de forma automática e completa.

Os utilizadores acedem ao Bug Hunt através de um *web browser*. Na primeira vez que é feito o acesso, os utilizadores são presenteados com instruções dos objetivos do tutorial e uma descrição de como usar a ferramenta. No caso de um estudante já ter iniciado as aulas é direcionado para a aula em que se encontrava quando o tutorial foi fechado pela última vez.

Durante o tutorial o aluno avança através de uma série de aulas, usando estratégias de teste específicas em cada aula de forma a encontrar erros em programas. O seu progresso vai sendo medido pelo número de falhas detetadas e pelo seu desempenho relativo aos seus colegas de turma.

Cada lição incorpora um conjunto de objetivos, um exercício e os resultados do exercício. Os objetivos são facultados ao aluno no início de cada aula, onde estão descritos os conceitos de teste com que o aluno deve estar familiarizado no final da aula e a estratégia de teste que será praticada. O exercício é composto por um conjunto de componentes:

- **Instruções** – descrevem o desafio da aula e as atividades de teste;
- **Artefactos** – fornecem a informação necessária para a resolução da aula;
- **Resultados imediatos** – adaptados de acordo com o tipo de teste que está a ser realizado;
- **Testes** – submetidos durante a aula, um de cada vez, na área de casos de teste;
- **Ajuda** – dicas para ajudar o aluno na resolução do exercício.

No final do exercício o estudante recebe os resultados face ao número de falhas encontradas, assim como informação adicional sobre as falhas descobertas. Isto ajuda o estudante a perceber os

conteúdos que reteve e aqueles que não conseguiu entender, para assim perceber em que conteúdos deve focar o seu estudo.

Assim que o tutorial está completo, o estudante recebe uma avaliação geral do seu desempenho. A figura 2.9 apresenta a estrutura do Bug Hunt pela perspetiva do aluno.

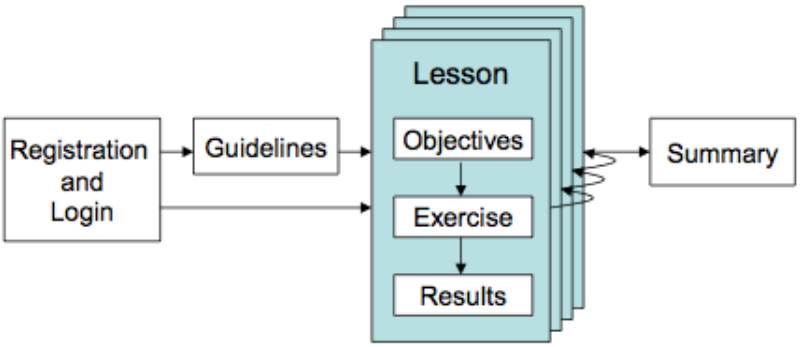


Figura 2.9: Estrutura de alto nível do tutorial Bug Hunt [20].

Esta estrutura baseada em quatro lições pretende aumentar gradualmente os conhecimentos dos alunos sobre as técnicas de teste de software. A primeira aula faz uma introdução básica aos conceitos de testes de software e à sua terminologia, onde o objetivo é encontrar uma falha particular através dos casos de teste que são criados com as pistas fornecidas. As aulas seguintes trabalham sobre estes conceitos, colocando o aluno numa perspetiva prática sobre as técnicas de caixa-preta e caixa-branca. Na última aula é feito um apanhado de tudo introduzindo o estudante aos testes automáticos. Todo o trabalho do estudante é gravado ao longo do tutorial para que este possa sair e voltar a entrar no Bug Hunt sem perder o seu trabalho.

Para o professor, o Bug Hunt fornece um relatório completo dos resultados de execução dos casos de teste dos seus estudantes. A informação apresentada incluí valores de entrada dos casos de teste, os valores esperados de saída, os valores de saída obtidos pelo estudante e o número da aula em que o aluno criou o caso de teste, assim como os resultados da execução dos casos de teste. Uma vez que os alunos estão agrupados em turmas o professor tem acesso ao desempenho do aluno face aos restantes. O professor, por contato com a equipa de desenvolvimento do Bug Hunt, tem a possibilidade de configurar as aulas para a sua turma, através de um conjunto de exercícios pré-definidos ou da proposta de novos exercícios.

2.4.2.3 TestEG

O TestEG [21] é um jogo educacional computacional, do tipo *quiz*, na área de testes de software capaz de minimizar os problemas encontrados no processo de ensino-aprendizagem de alguns assuntos abordados nessa área.

Neste jogo, em que o cenário de jogo é um ambiente empresarial (figura 2.10), o aluno assume o papel de gestor de testes de software. Assim que o jogo é iniciado, o jogador recebe o seu orçamento e deve contratar a sua equipa de três testadores. No decorrer do jogo, o gestor deve

ajudar a sua equipa na resolução de problemas e esclarecimento de dúvidas, dando as informações necessárias para a execução do seu trabalho. Na totalidade, o jogador deve responder a dez perguntas dentro do tempo de dez minutos, sem que o orçamento se esgote. Este poderá também realizar formações aos seus funcionários, verificar os seus desempenhos e ler conteúdos teóricos sobre testes de software. Quando é realizada uma formação a um funcionário é descontado do orçamento um valor relativo ao tipo de formação. No final do jogo o jogador pode visualizar a sua posição na tabela de pontuações geral dos jogadores.

Figura 2.10: Interface gráfica do jogo TestEG [21].

Este jogo também está preparado para professores. Estes têm como funções: registar os seus alunos e controlar o conteúdo a ser transmitido ao aluno, para que possa avaliá-lo e sentir quais são realmente as suas maiores dificuldades.

O objetivo deste jogo é tornar mais lúdico o processo de ensino-aprendizagem sobre testes de software, de uma forma mais atrativa e próxima do ambiente real de trabalho.

2.5 Resumo e Conclusões

Neste capítulo encontra-se documentado o trabalho de estudo realizado sobre o Estado da Arte (2) nos dias de hoje. O estudo baseou-se na procura de metodologias de desenvolvimento de jogos, não só no campo geral, mas também, especificamente no desenvolvimento de jogos educativos. As metodologias encontradas que se mostraram pertinentes, e que foram descritas, para a realização deste trabalho foram:

- 2.2.1 Metodologia ADDIE - processo de modelo para a criação de software, normalmente associado à criação de ferramentas de ensino supervisionadas por um professor. Esta é usada por muitas empresas de *design* e de desenvolvimento de *e-learning*;
- 2.2.2 Método Ágil SCRUM - método ágil que utiliza uma abordagem iterativa ideal para projetos complexos e imprevisíveis;
- 2.2.3 Metodologia Hybrid - metodologia criada a partir da fusão das metodologias ADDIE e SCRUM. Esta é capaz de se adaptar de acordo com o tamanho, o âmbito e o nível de complexidade de cada projeto;
- 2.2.4 Metodologia de Chandler - estrutura básica para o processo geral de produção de jogos sob a perspetiva do produtor;
- 2.2.5 Metodologia de Schuytema - abordagem para o desenvolvimento de jogos constituída por três grandes períodos no ciclo de desenvolvimento, pelo ponto de vista do *designer*.

Neste capítulo foi também feita uma revisão tecnológica às principais ferramentas de desenvolvimento de jogos que poderão ser utilizadas como forma de auxílio no desenvolvimento do projeto. Exemplos delas são: 2.3.1 Stencyl, 2.3.2 Construct 2, 2.3.3 Unity e 2.3.4 GameSalad. As características, vantagens e desvantagens da sua utilização foram discutidas nas respetivas secções.

Por fim, foi apresentado o estudo sobre os projetos existentes na literatura, tanto na área de engenharia de software como no domínio específico dos testes de software, a fim de estudar as diversas alternativas já existentes e como seria possível criar um projeto inovador nesta área.

Em resumo, nenhum dos jogos sérios estudados (U-TEST, Bug Hunt, TestEG) cobre o corpo de conhecimento exigido pelo nível base da certificação do ISTQB. Para além disso, estes jogos pressuponhem que os jogadores têm conhecimento básico de testes de software e, por isso, não explicam os conceitos antes de cada desafio.

O jogo U-TEST baseia-se no reconhecimento e compreensão dos conceitos principais de testes de software de uma forma geral e na perceção e aplicação das técnicas de seleção de entrada de dados, partição da classe de equivalência e análise do valor limite. O maior problema deste jogo é o facto de se focar apenas em testes unitários e nas técnicas de caixa-preta.

O jogo Bug Hunt é composto apenas por cinco aulas. A primeira aula faz uma introdução básica aos conceitos de testes de software e à sua terminologia, onde o objetivo é encontrar uma falha particular através dos casos de teste que são criados com as pistas fornecidas. As aulas seguintes trabalham sobre estes conceitos, colocando o aluno numa perspetiva prática sobre as técnicas de caixa-preta e caixa-branca. Na última aula é feito um apanhado de tudo introduzindo o estudante aos testes automáticos. Este jogo mostrou-se o mais completo dos três mas apenas permite ao jogador treinar testes de software de uma forma prática, pois foca-se na conceção de testes e na procura de falhas.

Já o jogo TestEG, apesar da sua interface gráfica, é um jogo do tipo *quiz* bastante simples, onde o jogador tem apenas de responder a perguntas de escolha múltipla.

O *iLearnTest* apresenta uma abordagem capaz de resolver/melhorar estes problemas. A abordagem desenhada abrange o corpo de conhecimento do nível base do ISTQB, para que os alunos que pretendem obter a certificação possam utilizar o *iLearnTest* como ferramenta de estudo. Para além disso, a abordagem não se limita à criação de testes nem a perguntas do tipo *quiz*, apresenta jogos/desafios para todas as matérias com um *design* moderno, para assim cativar os utilizadores para o estudo.

Relativamente às metodologias de desenvolvimento, a ADDIE é a metodologia mais completa e está especificamente desenhada para a criação de ferramentas para o ensino, sendo utilizada por diversas empresas de desenvolvimento de software educativo. As vantagens deste modelo refletem-se na gestão do tempo de desenvolvimento e nos custos do projeto, na eficácia da avaliação dos resultados e no êxito da aprendizagem transmitida.

Relativamente às tecnologias de suporte ao desenvolvido de jogos, constatou-se que o Construct 2, oferece certas características que sobressaem em relação às outras tecnologias apresentadas, como o facto de permitir exportar o projeto para *HTML5* e ter acesso ao código, podendo assim ser utlizado na construção de um *website*.

Capítulo 3

iLearnTest

Este capítulo apresenta o jogo educacional *iLearnTest*, cujo objetivo é apoiar o processo de ensino/aprendizagem de teste de software para quem pretende realizar o exame de obtenção do nível base de certificação do ISTQB. Para tal, detalhada toda a implementação, descreve os objetivos de aprendizagem e a estrutura do jogo. Inicialmente, a secção 3.1 faz uma introdução ao jogo. A secção 3.4 expõem o *website* construído para o iLearnTest. A secção 3.2 descreve a implementação do *iLearnTest*. A secção 3.3 expõem os objetivos de aprendizagem que o jogo pretende transmitir. A secção 3.5 explica a estrutura do jogo e os desafios que o constituem. Por fim, a secção 3.6 apresenta um resumo e as conclusões do capítulo.

3.1 Introdução

O ensino baseado em jogos tem um grande potencial no aumento da motivação e envolvimento dos estudantes para apreensão de novos conceitos, como comprovam estudos existentes na literatura [5, 6, 7, 8, 9, 10]. O *iLearnTest* vem facilitar a aprendizagem e o treino dos conhecimentos na área de testes de software, captando a atenção, motivando e incentivando o envolvimento dos estudantes.

O *iLearnTest* não pretende substituir o ensino tradicional, mas apresentar-se como uma opção complementar ao ensino de testes de software e como uma forma de ajudar a aumentar o interesse dos estudantes para o tema. Para além disto, o *iLearnTest* tem como objetivo ajudar os estudantes na preparação para a realização do exame de certificação *Foundation* do ISTQB (*International software Testing Qualification Board*). Este é o primeiro nível do programa de certificação internacional em testes de software.

A qualificação de nível *Foundation* destina-se a qualquer indivíduo envolvido em testes de software, independentemente da função desempenhada, tais como: testadores (*testers*), analistas de testes, engenheiros de testes, consultores de testes, gestores de testes, testadores (*testers*) de aceitação de utilizador e programadores. A qualificação de nível *Foundation* é também sugerida

para quem quer obter conhecimentos básicos em testes de software, tais como gestores de projeto, gestores de qualidade, gestores de desenvolvimento de software, analistas de negócio, diretores de tecnologias da informação e consultores de gestão. Os detentores da certificação de nível *Foundation* poderão, posteriormente, evoluir para uma qualificação de nível superior em testes de software [22].

O corpo de conhecimento (*Syllabus*) para esta certificação é estruturado em seis capítulos:

- Fundamentos de testes;
- Testes através do Ciclo de Vida de software;
- Técnicas Estáticas;
- Técnicas de Conceção de testes;
- Gestão de testes;
- Ferramentas de Suporte aos testes.

O *iLearnTest* contém várias características que o tornam atraente para os alunos. Para além de transmitir os conhecimentos atrás mencionados, pretendia-se que o *iLearnTest*:

- Incorporasse desafios em cada matéria para promover o envolvimento dos estudantes durante a prática dos conhecimentos e técnicas de teste de software;
- Oferecesse a possibilidade de estudo individual para que os alunos pudessem aprender ao seu próprio ritmo, gastando o tempo que achassem necessário para a aprendizagem;
- Atribuísse uma pontuação em cada jogo concluído pelo estudante, de forma a incentivar a obtenção da pontuação máxima e, por conseguinte, aumentar o seu conhecimento;
- Apresentasse os resultados dos estudantes após cada desafio, indicando as respostas certas e erradas e apresentando a correção neste último caso.

3.2 Implementação

O *iLearnTest* foi desenvolvido de acordo com a metodologia ADDIE. Esta metodologia envolve uma abordagem linear constituída por cinco etapas individuais e semi-discretas: análise, *design*, desenvolvimento, implementação e avaliação.

Na fase de análise do projeto foi definido o problema, o tipo de aprendizagem a transmitir e as matérias de testes de software que se pretendiam abranger. Nesta fase foi identificado o tipo de aprendizes a que o *iLearnTest* se destinava e analisado o seu conhecimento atual nesta área. Assim como as tecnologias disponíveis para o desenvolvimento e os projetos existentes na literatura, não só na área de testes de software como também de engenharia de software.

Uma vez completa a fase de análise, o passo seguinte foi o *design* da solução. Os objetivos da aprendizagem foram especificados, assim como a estratégia de ensino a adotar. A criação da aprendizagem e das interfaces necessárias para a ferramenta aconteceu na etapa de desenvolvimento. De seguida aconteceu a fase de implementação, onde se realizou a programação do jogo. Por fim, foi elaborada a validação com utilizadores reais, onde se analisou a utilidade da ferramenta de aprendizagem construída.

O *iLearnTest* é implementado com base na *web*, o que significa que os utilizadores só necessitam de um *browser* para jogar. A plataforma escolhida para o seu desenvolvimento foi a Construct 2 [15], já que esta oferece certas características que sobressaem em relação às outras tecnologias estudadas. Este software oferece a possibilidade de desenvolvimento de jogos 2D em *HTML5* através de um sistema de *drag-and-drop*, um editor visual e um sistema de lógica baseada em comportamentos. Este sistema oferece comportamentos flexíveis, pré-visualização instantânea, efeitos visuais aliciantes, publicação em todas as plataformas e fácil extensibilidade através da agregação de diversos *plugins*. Após a importação do projeto a ferramenta disponibiliza o código gerado possibilitando ao programador embutir o jogo num *website*.

Já num estado avançado do processo de implementação foi atingido o número limite de eventos que a versão grátis do Construct 2 [15] disponibiliza, não permitindo a continuação do projeto, por consequência foi necessário proceder à compra da versão paga que não tem quais quer limitações.

Implementou-se também o carregamento de conteúdos para o *iLearnTest* através de um XML, de forma a facilitar a posterior atualização de conteúdos. Este XML contém os títulos dos capítulos e suas secções, os objetivos de aprendizagem de cada secção e, também, os conteúdos teóricos de cada uma.

3.3 Objetivos de Aprendizagem

O método de ensino do *iLearnTest* baseia-se no *Syllabus* (corpo de conhecimento) do programa de certificação base para a Qualificação Internacional em Testes de Software no nível *Foundation*. Este programa encontra-se organizado em seis capítulos principais, dentro de cada capítulo existem várias secções. Inicialmente cada secção apresenta os seus objetivos de aprendizagem (LOs) que identificam o que o formando deverá saber após a conclusão de cada módulo.

Uma vez que o *Syllabus* é bastante extenso, selecionou-se, de entre todas, cinco secções, do capítulo três e quatro. O desenvolvimento do jogo para estes capítulos é uma prova de conceito que depois se poderá estender de forma a cobrir as outras matérias do corpo de conhecimento do ISTQB. A fim de criar desafios mais aliciantes e de maior importância para a formação dos estudantes escolheram-se as secções que apresentam maior relevância na área de testes de software e que se baseiam não só em conceitos teóricos como práticos. De seguida apresentam-se os objetivos de aprendizagem do *Syllabus* do ISTQB que o *iLearnTest* cobre atualmente:

- **3.2. Processo de Revisão**

LO-3.2.1. Recordar as atividades, funções e responsabilidades de uma revisão formal típica.

- **3.3. Análise Estática via Ferramentas**

  LO-3.3.1. Recordar os defeitos e erros típicos detetados pela análise estática e compará-los com as revisões e testes dinâmicos.

  LO-3.3.2. Descrever, por exemplos, os benefícios típicos da análise estática.

  LO-3.3.3. Listar os defeitos típicos de código e concepção que podem ser detetados pelas ferramentas de análise estática.

- **4.2. Categorias das Técnicas de concepção de testes**

  LO-4.2.1. Recordar as razões pelas quais as técnicas de concepção de testes, baseadas nas especificações (caixa-preta) e baseadas na estrutura (caixa-branca), são úteis e listar as técnicas comuns a ambas.

  LO-4.2.2. Explicar as características, semelhanças e diferenças entre testes baseados nas especificações, testes baseados na estrutura e testes baseados na experiência.

- **4.3. Técnicas Baseadas nas Especificações ou Caixa-Preta**

  LO-4.3.1. Escrever casos de teste com base em modelos de software fornecidos utilizando a partição por equivalências, a análise de valor fronteira, tabelas de decisão e diagramas/tabelas de transição de estados.

  LO-4.3.2. Explicar o objetivo principal de cada uma das quatro técnicas de teste, qual o nível e tipo de teste que pode utilizar cada técnica, e de que forma pudemos medir a cobertura.

  LO-4.3.3. Explicar o conceito de testes de casos de uso e os seus benefícios.

- **4.4 Técnicas Baseadas na Estrutura ou Caixa-Branca**

  LO-4.4.1. Descrever o conceito e valor da cobertura de código.

  LO-4.4.2. Explicar os conceitos de cobertura de instruções ou decisões, fundamentando a sua utilização a outros níveis de teste que não os testes de componentes (p. ex. ao nível de teste de sistema em procedimentos de negócio).

  LO-4.4.3. Escrever casos de teste a partir de fluxos de controlo utilizando as técnicas de concepção de testes por instrução ou decisão.

  LO-4.4.4. Avaliar a cobertura de instruções e decisões face ao comprimento integral de critérios de saída definidos.

3.4 O *website* do *iLearnTest*

Nos dias de hoje, em que a Internet é o ponto máximo de difusão da informação, um simples acesso a um dispositivo com rede permite chegar a toda esta informação em qualquer local a qualquer hora. Por isso tornava-se indispensável criar uma forma simples e fácil para os alunos utilizarem o *iLearnTest* através desta rede. Como tal, criou-se um *website* que permite aos utilizadores aprender com o *iLearnTest* quando mais desejarem.

O *website* encontra-se estruturado em cinco páginas:

- **Página de entrada** - o utilizador tem a oportunidade de entrar no *website* através de autenticação com o seu nome e palavra-passe (figura 3.4);
- **Página inicial** - o utilizador tem a oportunidade de jogar o *iLearnTest* B.2;
- **Página de pontuações** - tabela com as melhores pontuações dos utilizadores registados (figura B.3);
- **Página de informação** - contém informação sobre o propósito do *iLearnTest*;
- **Página de contacto** - permite ao utilizador entrar em contacto, através de *email*, para esclarecer alguma dúvida ou fazer sugestões.

Em relação às tecnologias utilizadas para a sua implementação escolheu-se como linguagem de escrita *HTML5*, uma vez que esta é a linguagem mais comum utilizada na produção de *websites* e é reconhecida por todos os *browsers*. Para definir os estilos das páginas utilizou-se *CSS*, a linguagem mais utilizada em conjunto com *HMTL5* para tal. Utilizou-se *JavaScript*, juntamente com a biblioteca *jQuery*, como linguagem de programação para a implementação de conteúdo dinâmico do lado do cliente, esta é atualmente a linguagem principal para este fim. Para facilitar e simplificar a transmissão de dados entre o servidor e as páginas utilizou-se o formato *JSON*. Do lado do servidor utilizou-se a linguagem *PHP* pois permite uma simples e eficaz ligação entre o lado do cliente e a base de dados, uma vez que a base de dados necessária para o *website* é simples e de pequena dimensão utilizou-se *SQLite*.

3.5 Estrutura do Jogo

O *iLearnTest* apresenta uma estrutura de menus compatível com a estrutura de capítulos do *Syllabus Foundation* do ISTQB (versão de 2011).

A navegação pelos menus é baseada num jogo de plataformas, em que o personagem é controlado pelas teclas de setas do teclado. Para navegar pelos menus o utilizador necessita de colocar o personagem em cima das plataformas, com a forma de quadrados e retângulos, e clicar na tecla "Seta para Baixo"do teclado. Por consequência, o personagem submerge e é levado a cair numa outra página.

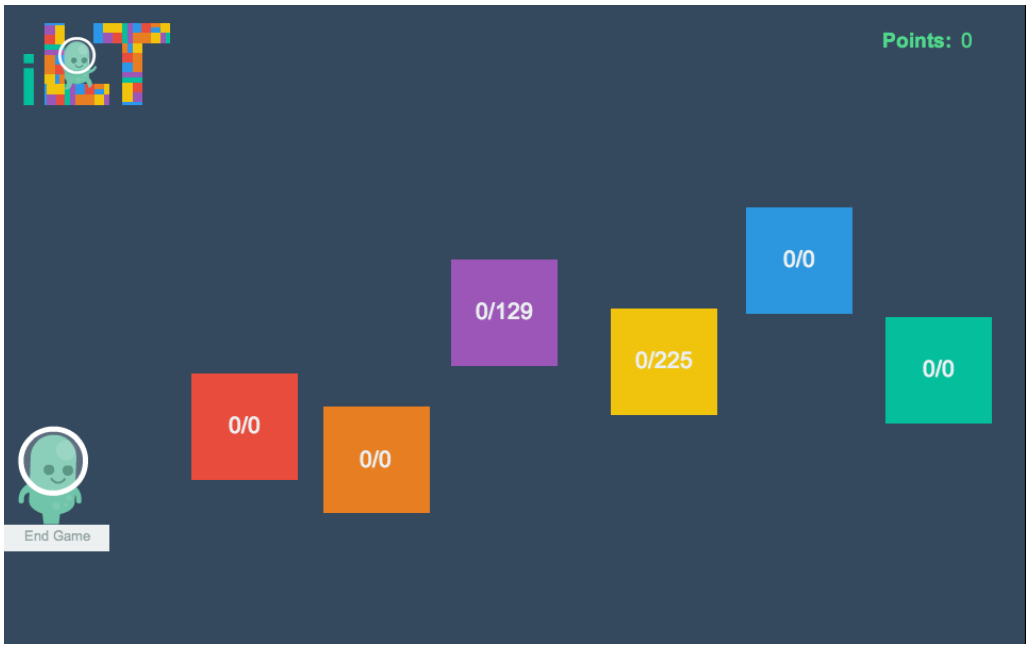


Figura 3.1: Menu inicial do *iLearnTest*

A figura 3.1 apresenta o primeiro menu, que é exibido após a página de início. Como é possível visualizar, este menu é constituído por seis plataformas coloridas, em que cada uma representa um dos seis grandes capítulos do *Syllabus*:

1. Fundamentos de testes;
2. Testes através do Ciclo de Vida de software;
3. Técnicas Estáticas;
4. Técnicas de conceção de testes;
5. Gestão de testes;
6. Ferramentas de Suporte aos testes.

Ao colocar o personagem sobre uma plataforma é exibido um balão com o título do capítulo correspondente (figura 3.2). Cada plataforma apresenta, também, a pontuação obtida pelo utilizador até ao momento (numerador) versus o total de pontos que o jogador poderá obter jogando todos os desafios de cada matéria (denominador). A plataforma branca está presente em todas as páginas para possibilitar ao jogador voltar à página anterior, ou no caso do menu inicial (figura 3.1) terminar o jogo.

Esta versão do *iLearnTest* apresenta o capítulo três e três secções do capítulo quatro cobertas, como é possível perceber pela imagem do menu inicial 3.1, pois tanto quanto se pode observar as plataformas correspondentes são as únicas que contêm a pontuação total possível de obter (denominador) superior a zero.

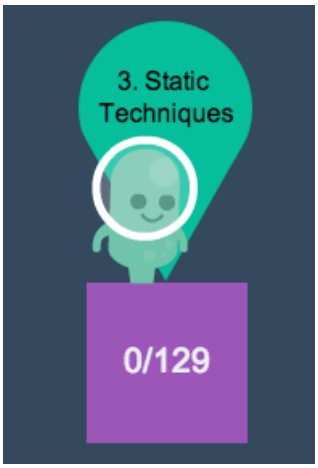


Figura 3.2: Exemplo de balão com título de capítulo.

Dentro de cada secção, inicialmente são apresentados os objetivos de aprendizagem (figura 3.3) dessa parte, depois é exibida a matéria teórica importante ao jogo/desafio que se realiza de seguida, em que os estudantes podem exercitar esses conceitos teóricos.

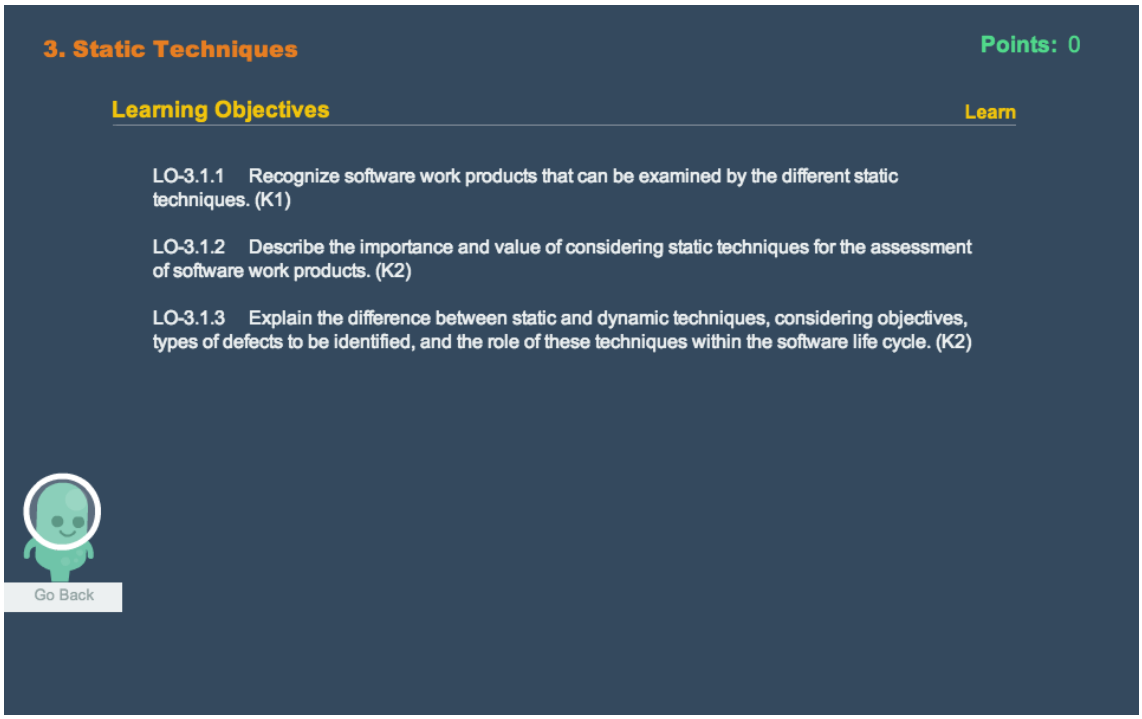


Figura 3.3: Exemplo de página com os objetivos de aprendizagem da secção.

Existem diferentes jogos para diferentes matérias. De seguida são descritos alguns exemplos de jogos presentes no *iLearnTest*.

3.5.1 Adivinhar os Conceitos

Neste jogo, do tipo jogo da forca, pretende-se que o jogador consiga identificar os conceitos que são descritos textualmente. No caso do jogador se encontrar com dificuldade, em encontrar o conceito, tem a hipótese de utilizar o botão de ajuda, que lhe dará uma letra da palavra, sendo que, depois a pontuação a obter por acertar a palavra será menor.

No contexto do programa de certificação do ISTQB este jogo foi utilizado para ensinar as diferentes atividades de uma revisão formal.

Os diferentes tipos de revisões variam desde a informal, caracterizada pela falta de instruções escritas para os revisores, até à sistemática, caracterizada pela inclusão da participação da equipa, documentação dos resultados da revisão e procedimentos para a realização da revisão. A formalidade de um processo de revisão está relacionada com fatores como a maturidade do processo de desenvolvimento, requisitos legais ou regulamentares ou a necessidade de um caminho de auditoria. A maneira como a revisão é realizada depende dos objetivos acordados da revisão [22].

Uma revisão formal típica tem as seguintes atividades principais: Planeamento, *Kick-off*, Preparação individual, Reunião de revisão, Refazer o trabalho, *Follow-up*. Tendo sido estas as atividades usadas para o jogo em questão.

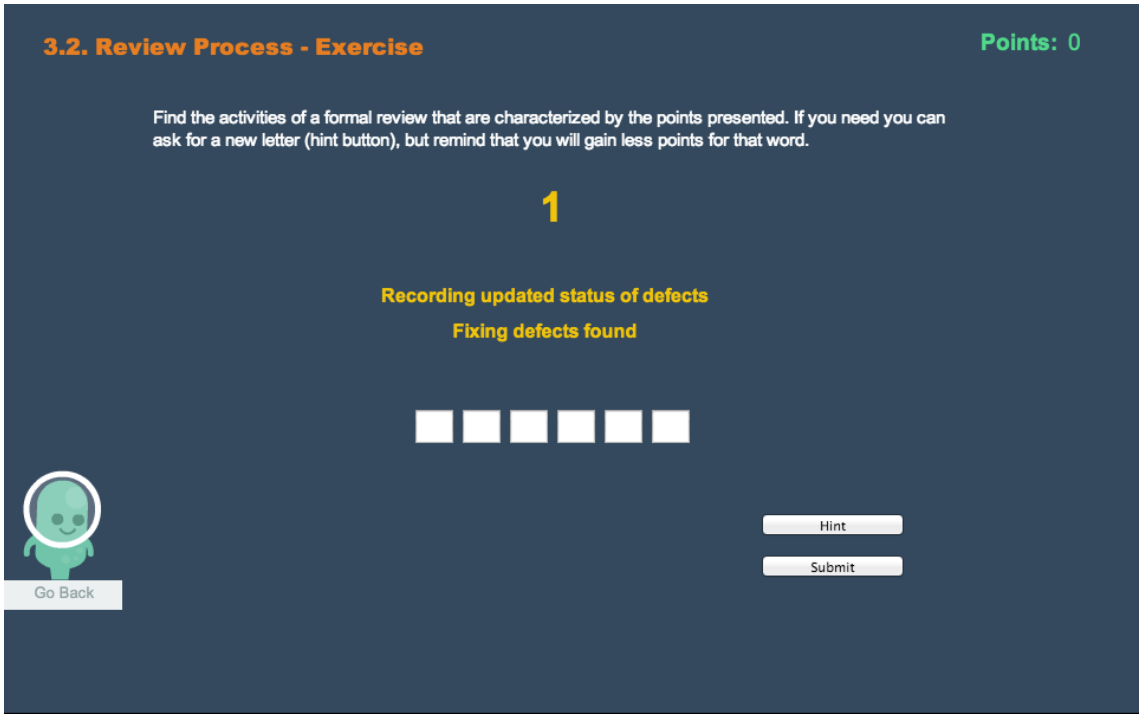


Figura 3.4: Imagem do jogo *Adivinhar Conceitos* (3.5.1).

3.5.2 *Drag-and-drop* para Duas Caixas

Este jogo pede ao jogador que junte conceitos em grupos. Através de *drag-and-drop*, os conceitos, que se encontram desorganizados pelo ecrã, devem ser arrastados para dentro da respetiva caixa de agrupamento.

Na área de testes de software é importante perceber a diferença entre técnicas de conceção de testes de caixa-preta e de caixa-branca.

Conforme [22], as técnicas de conceção de testes de caixa-preta são realizadas com base na análise da documentação que serve de base para testes, sendo assim possível derivar e selecionar as condições de teste, casos de teste, ou dados de teste. Os testes de caixa-preta, por definição, não utilizam qualquer informação sobre a estrutura interna do componente ou sistema a ser testado.

As técnicas de conceção de testes baseados em caixa-branca consistem na análise da estrutura em três distintos níveis:

- Ao nível do componente: a estrutura de um componente de software, ou seja, as suas instruções, decisões, ramos ou caminhos distintos;
- Ao nível da integração: a estrutura pode ser uma árvore de chamada (um diagrama no qual módulos invocam outros módulos);
- Ao nível do sistema: a estrutura pode ser um menu estruturado, um processo de negócio ou a estrutura de uma página web.

Desenvolveu-se este jogo para que o jogador aprenda quais as técnicas de conceção de testes de software que são de caixa-preta e quais as que são de caixa-branca. O agrupamento das técnicas é realizado, através de *drag-and-drop*, para uma caixa preta e uma caixa branca.

3.5.3 Apanhar os Conceitos Corretos

Neste jogo o utilizador tem de movimentar o personagem, com as setas do seu teclado, por baixo de conceitos que se encontram a cair, apanhando os conceitos corretos e deixando cair os que não estejam corretos.

Segundo [22], os testes podem ser originados de casos de uso. Um caso de uso descreve as interações entre os atores (utilizadores ou sistemas), que produzem um resultado de valor para o utilizador do sistema ou para o cliente. Os casos de uso descrevem os "fluxos de processo" através de um sistema baseado na sua utilização atual real. Os casos de teste derivados de casos de uso são mais úteis na deteção de defeitos no processo de fluxos durante a utilização real do sistema.

Por tanto, para ensinar casos de uso, o *iLearnTest* faz inicialmente uma introdução teórica aos casos de uso e de seguida permite que o jogador jogue o *Apanhar os Conceitos Corretos*. Nele pretende-se que o utilizador apanhe apenas os conceitos relativos à interação do utilizador com o sistema, segundo o diagrama que é apresentado inicialmente.

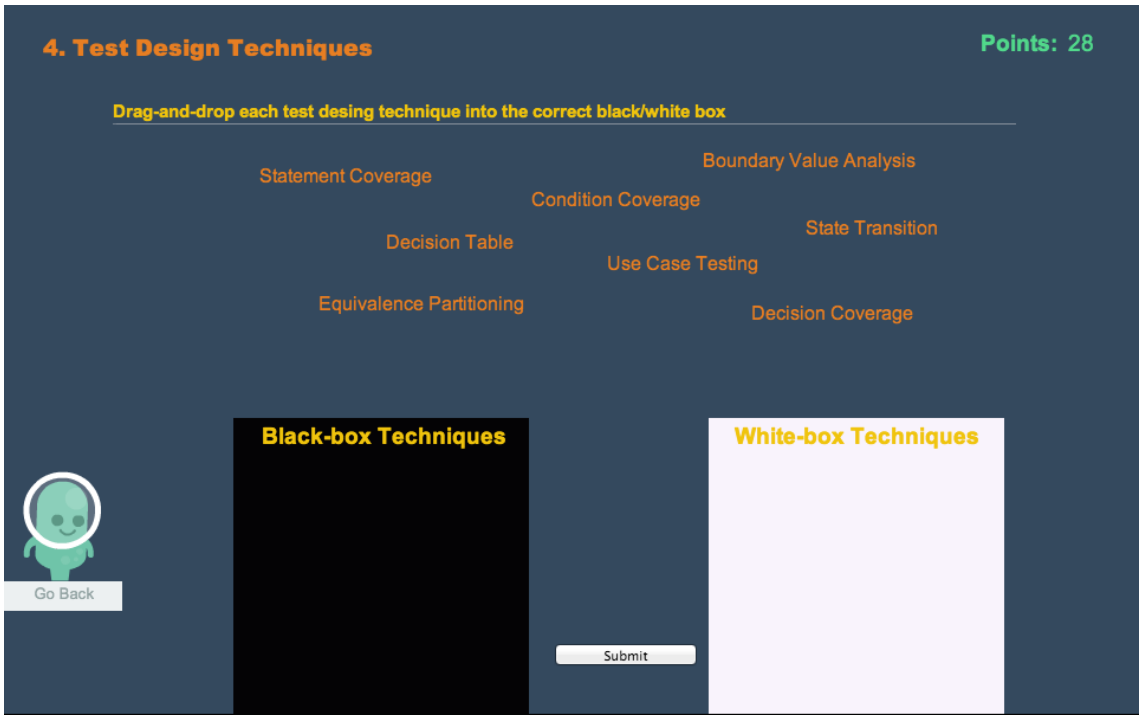


Figura 3.5: Imagem do jogo *Drag-and-drop para Duas Caixas* (3.5.2).

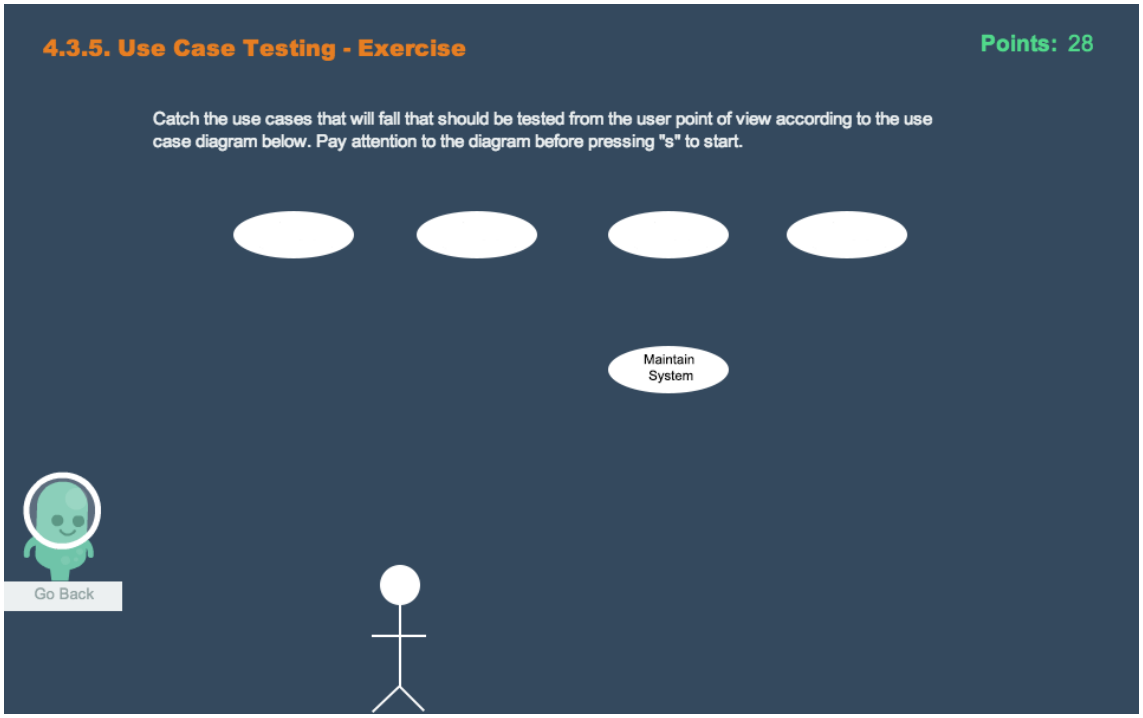


Figura 3.6: Imagem do jogo *Apanhar os Conceitos Corretos* (3.5.3).

3.5.4 Descobrir os Caminhos

Neste jogo é apresentado um grafo no qual o jogador deve selecionar, com o rato, os componentes (caixas e setas) que achar necessários para a criação do caminho pedido no enunciado.

Optou-se, para ensinar os conceitos relativos a cobertura de testes, por desenhar o grafo de fluxo de controlo e pedir ao utilizador para selecionar nesse grafo os elementos que correspondiam a uma cobertura de 100% de instruções, 100% de decisões. Relativamente à cobertura de condições pede-se que se indique os casos de teste (valores para as diversas variáveis envolvidas) que permitem atingir 100% de cobertura.

Considera-se a cobertura de testes como a medida em que uma estrutura é executada por um conjunto de testes, expressa em percentagem dos itens a serem cobertos. Se a cobertura não atingir os 100%, então mais testes devem ser concebidos para testar os itens que não foram abrangidos e assim aumentar a cobertura [22].

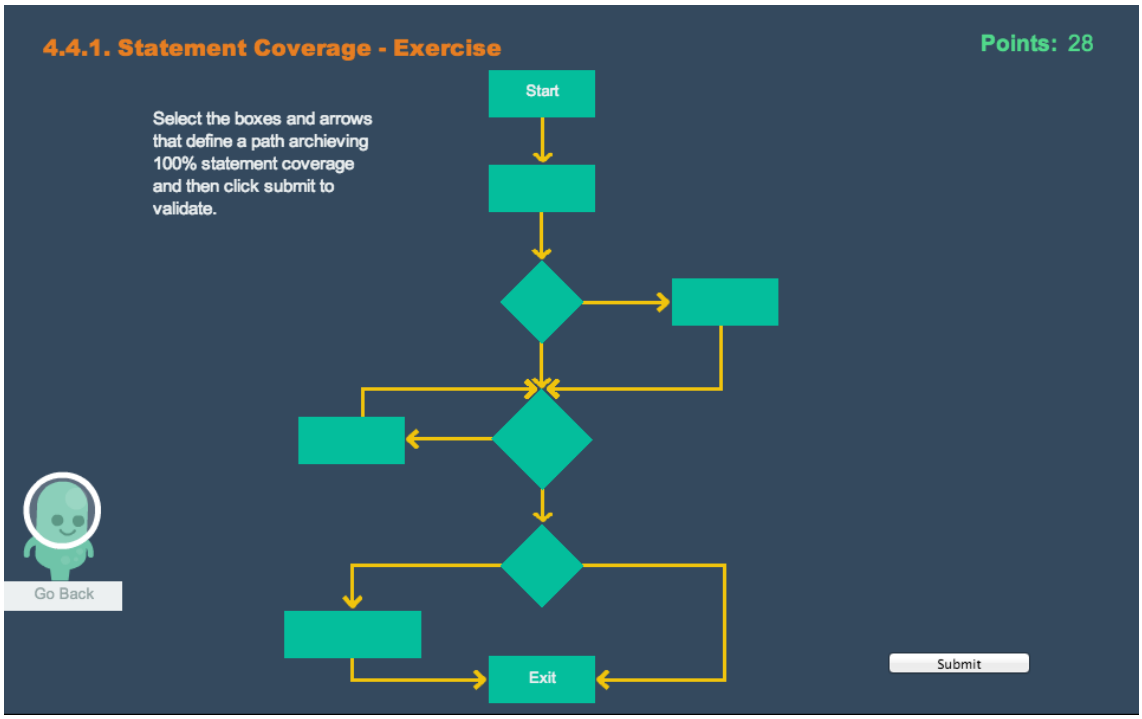


Figura 3.7: Imagem do jogo *Descobrir os Caminhos* (3.5.4).

Caso o jogador não atinja a pontuação máxima em cada jogo, tem a hipótese de voltar a jogar quantas mais vezes necessitar. Esta procura pela pontuação máxima fará com que o aluno entre em contacto um maior número de vezes com os conceitos que estão a ser ensinados e, por consequência, aumente o seu nível de aprendizagem.

No final de cada capítulo o jogador tem a possibilidade de realizar um mini *quiz* de seis questões, relacionadas com a matéria ensinada no capítulo, do tipo das questões de escolha múltipla do

exame de certificação de testes de software *Foundation* do ISTQB. A figura 3.8 apresenta , como exemplo, o *quiz* realizado no final do capítulo 4. Nela é possível perceber que o jogador deverá responder às seis perguntas escolhendo a resposta certa de entre as escolhadas existentes em cada caixa. Depois de submetida a resposta o *iLearnTest* indica quais as respostas certas e quais as erradas, como mostra a figura 3.8.

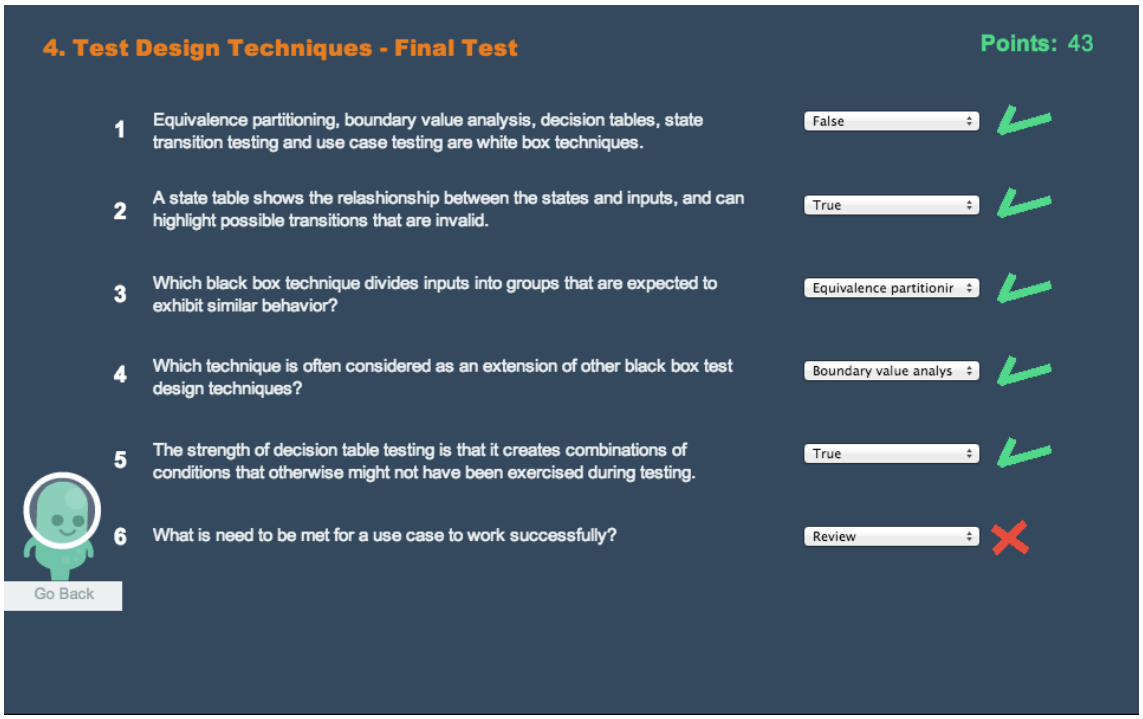


Figura 3.8: Exemplo de *quiz* do final de um capítulo.

3.6 Resumo e Conclusões

Este capítulo faz uma revisão ao *iLearnTest* e aos seus objetivos na secção 3.1. A secção 3.2 descreve o processo de implementação do jogo educacional desenvolvido. A secção seguinte, 3.3, descreve os objetivos de aprendizagem que são cobertos pela versão atual do *iLearnTest*. A secção 3.4 detalha o *website* implementado. A secção 3.5 documenta toda a estrutura do jogo, assim como os jogos/desafios implementados:

- *3.5.1 Adivinhar os Conceitos* - o jogador deve adivinhar os conceitos que são descritos pelos pontos apresentados;
- *3.5.2 Drag-and-drop para Duas Caixas* - o jogador deve organizar os conceitos apresentados no ecrã em duas caixas, de acordo com a sua categoria;
- *3.5.3 Apanhar os Conceitos Corretos* - o jogador tem de manipular o personagem para apanhar os conceitos corretos, deixando cair os incorretos;

- *3.5.4 Descobrir os Caminhos* - o jogador deve selecionar o caminho correto de um grafo de acordo com o pedido inicialmente.

Capítulo 4

Validação

Com o objetivo de avaliar a eficácia do *iLearnTest* no processo de ensino-aprendizagem de testes de software realizou-se a experiência descrita neste capítulo. A secção 4.1 faz uma introdução inicial à experiência. A secção 4.2 apresenta a hipótese à qual se pretende dar resposta. A secção 4.3 descreve o procedimento de toda a experiência. A secção 4.4 descreve os resultados alcançados com a experiência realizada. Por fim, a secção 4.5 expõem o resumo e as conclusões de todo o capítulo.

4.1 Introdução

A validação de um novo jogo de ensino é um ponto fulcral após a sua implementação. Testar e verificar a capacidade de transmissão dos conceitos é extremamente importante. Os utilizadores futuros devem ser capazes de utilizar o sistema, conseguir entender a sua estrutura e principalmente perceber os conteúdos explicados e os desafios que têm de enfrentar.

A fim de proceder a essa avaliação ao *iLearnTest*, e assim validar a abordagem, foi realizada uma experiência com doze alunos do quinto ano do Mestrado Integrado em Engenharia Informática e Computação, da Faculdade de Engenharia da Universidade do Porto, com idades compreendidas entre os 22 e os 24 anos, sem conhecimentos de testes de software.

Quatro dos alunos, selecionados aleatoriamente, foram utilizados como grupo de controlo, para servir de termo de comparação com o grupo experimental constituído pelos restantes oito alunos. Apenas o grupo experimental teve hipótese de utilizar o *iLearnTest* no teste à aprendizagem, para assim ser analisado se este é uma mais valia, ou não, no processo de ensino-aprendizagem de testes de software.

De salientar que esta experiência foi realizada com grupos de apenas quatro alunos pois tornou-se complicado encontrar alunos com tempo disponível.

4.2 Hipótese

Com a realização desta experiência pretendeu-se validar a abordagem aplicada na resolução do problema em estudo - aumento da qualidade de aprendizagem dos estudantes no estudo dos conteúdos de testes de software para a realização do exame de obtenção do nível base da certificação do ISTQB.

Para tal, a experimentação foi elaborada de modo a verificar a possibilidade, ou não, dos utilizadores aprenderem os conceitos teóricos exigidos pelos objetivos de aprendizagem (LO's), referidos a cima, contidos no *Syllabus* do ISTQB, com a exclusiva utilização do *iLearnTest*.

O ponto fulcral desta experiência incide sobre a resposta à questão:

Será o iLearnTest eficaz na transmissão de conhecimentos para a obtenção da certificação Foundation do ISTQB?

4.3 Procedimento

De forma a testar e validar o *iLearnTest* desenhou-se uma experiência com doze alunos sem conhecimentos de testes de software, visto que não frequentaram a unidade curricular de testes e qualidade de software e, por consequência, não possuem frequência nesta disciplina. Todos eles são estudantes do quinto ano do Mestrado Integrado em Engenharia Informática e Computação, da Faculdade de Engenharia da Universidade do Porto, com idades compreendidas entre os 22 e os 24 anos.

Esta experiência foi dividida em duas partes. A primeira serviu como teste ao jogo desenvolvido, como forma de encontrar falhas e erros e perceber se existiam conceitos que não estariam a ser bem transmitidos aos estudantes. Com a segunda parte da experiência pretendeu-se, então, validar a abordagem.

O método experimental elaborado baseia-se num estudo comparativo entre os conhecimentos de testes de software adquiridos após o estudo dos alunos que utilizaram o *iLearnTest* e os que não o fizeram. Para tal, os doze alunos foram aleatoriamente divididos em três grupos (grupo A, grupo B e grupo C) de quatro elementos cada. O grupo A tinha apenas a oportunidade de estudar os conceitos de testes de software através do *Syllabus* do ISTQB e da *web*, não tendo a possibilidade de utilizar o *iLearnTest* para a aprendizagem e preparação nesta matéria. Por outro lado, o grupo B e o grupo C tiveram apenas direito à utilização do *iLearnTest* para realizar o estudo e a preparação para o exame que, todos os grupos, realizaram no final da experiência semelhante aos exames de certificação do ISTQB. Para além destes materiais todos os grupos receberam uma lista com os objetivos de aprendizagem (referidos anteriormente) a serem testados no exame a que foram submetidos no final da experiência.

Após a divisão aleatória realizada constatou-se que os elementos do grupo A, a quem não foi providenciado o *iLearnTest*, possuíam, todos eles, maior média de curso do que cada elemento dos outros grupos (B e C). A média das médias de curso dos estudantes dos grupos foi calculada. O

grupo A apresenta maior média e o grupo C a menor média. Descobriu-se uma diferença de um valor e meio entre o grupo A e o B, e uma diferença de dois valores e quatro décimas entre o grupo A e o C.

A experiência dividiu-se em duas partes: a primeira realizou-se apenas com o grupo A e B, e a segunda apenas com o grupo C. O grupo A serviu como grupo de comparação de resultados não só para a primeira parte como para a segunda, não se mostrou necessário a utilização de um outro grupo, uma vez que todos os grupos foram submetidos às mesmas variáveis e este grupo servia para comparação de resultados.

Relativo à primeira parte da experiência, inicialmente explicou-se o procedimento a adotar durante a experiência assim como o seu objetivo a todos os participantes: validar, ou invalidar, o método de aprendizagem criado com o *iLearnTest* através da comparação dos resultados obtidos no exame final. Sendo, então, necessário a separação dos grupos e a diferença de materiais que cada um receberia para o estudo dos conceitos de testes de software que necessitavam de aprender. Depois foi apresentado, a todos os participantes, o *iLearnTest* com uma pequena demonstração do seu funcionamento.

Posteriormente deu-se início à experiência, em que os estudantes se encontravam separados pelos seus locais de trabalho, onde cada um possuía um computador. As folhas com os objectivos de aprendizagem referidos foram distribuídas dando-se início ao tempo de estudo, uma hora e quarenta e cinco minutos (1:45min). Durante este tempo o estudo dos grupos foi observado, reparando-se que o grupo A (sem *iLearnTest*) estava com maior concentração, realizou apontamentos dos tópicos considerados mais importantes, leu toda a matéria indicada do *Syllabus* e ainda realizou exames de exemplo de certificação do ISTQB encontrados na *web*. Por outro lado o grupo B encontrava-se mais animado no teste do jogo perdendo alguns minutos iniciais no conhecimento do funcionamento do jogo e divertindo-se com o personagem do jogo, só depois realizaram todos os jogos/desafios das diferentes matérias.

Concluído o tempo de aprendizagem os materiais de estudo foram recolhidos e foi distribuído o exame para avaliar os conhecimentos aprendidos pelos alunos. O exame foi constituído por perguntas de escolha múltipla, semelhante aos exames de certificação do ISTQB, abrangendo os objetivos de aprendizagem que estiveram em estudo. Durante trinta minutos os participantes tiveram de responder a vinte e uma perguntas, tanto teóricas como práticas.

No final os exames foram recolhidos, corrigidos e analisaram-se os resultados.

Uma vez que esta primeira parte serviu, não só como validação da capacidade de ensino, mas também como teste à plataforma, para a descoberta de erros e falhas no *iLearnTest*, foi necessário proceder à segunda parte da experiência. O procedimento da segunda parte foi igual ao da primeira parte (descrito em cima), com a exceção que desta vez o teste foi realizado apenas com o grupo C.

A análise dos resultados de ambas as partes é descrita na secção seguinte.

4.4 Resultados

Inicialmente foram analisados os resultados da primeira parte da experiência. Na tabela da figura 4.1 é possível observar a comparação dos resultados obtidos pelos participantes desta parte, grupo A e B. As perguntas que cada elemento teve corretas estão assinaladas com um (1) e as erradas assinaladas com zero (0).

			Perguntas																					
Grupo	Aluno	Média	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Resultado	Nota
A	1	15,71	1	1	1	1	1	0	1	0	1	1	0	1	1	1	1	1	1	0	1	0	1	16	16,80
	2		1	0	0	0	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	16	16,80
	3		1	1	0	1	1	1	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	17	17,85
	4		0	1	0	1	0	0	1	0	1	0	1	1	1	0	0	1	1	1	1	1	1	13	13,65
		Total:	3	3	1	3	3	2	4	1	3	2	3	4	3	2	3	4	4	3	4	3	4	15,50	16,28
B	5	14,26	0	1	1	0	1	0	1	0	0	1	1	1	1	1	1	1	0	0	1	0	0	12	12,60
	6		0	1	0	1	1	1	0	0	0	0	1	1	1	0	1	1	0	0	0	1	0	10	10,50
	7		0	1	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1	1	0	1	1	15	15,75
	8		1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	0	1	1	0	1	17	17,85
		Total:	1	4	2	3	4	2	3	1	0	3	4	4	3	3	4	4	1	2	2	2	2	13,50	14,18
		Total dos Grupos:	4	7	3	6	7	4	7	2	3	5	7	8	6	5	7	8	5	5	6	5	6		

Legenda de cores:
Melhor resultado com iLT
Melhor resultado sem iLT
Todos os elementos acertaram
1 Elemento errou
2 Elementos erraram

Figura 4.1: Tabela de resultados da primeira parte da experiência.

O grupo A, o qual não utilizou a plataforma, apresenta uma média geral de curso de 15,71 valores. Já o grupo B, que utilizou o *iLearnTest* apresenta uma média de 14,26 valores. O primeiro atingiu uma média de notas no exame realizado de 16,28 valores. Por outro lado, o grupo B, que apresenta uma média geral de curso de menos 1,45 valores, atingiu uma média no exame de 14,18 valores.

Os resultados obtidos nesta primeira fase da experiência foram analisados no sentido de perceber quais as áreas em que os alunos tinham mais falhas e tentar melhorar o *iLearnTest* para ultrapassar essas dificuldades.

A cor amarela ressalta as perguntas em que houve maior número de alunos a acertar no grupo que utilizou *iLearnTest*. Já a cor cinzenta ressalta as perguntas em que houve maior número de elementos a acertar do grupo sem a plataforma. A cor verde indica que todos os elementos do grupo acertaram a pergunta. Por outro lado, a cor vermelha mais forte indica que apenas um acertou e a cor vermelha menos forte que apenas dois tiveram correta a pergunta.

A coluna *Resultado* indica o somatório de respostas certas por aluno. Já a coluna *Nota* indica a nota do aluno no exame, numa escala de zero a vinte. Analisou-se também o número de alunos total que acertaram cada pergunta, como é possível verificar na última linha da tabela da figura 4.1.

A partir desta análise fez-se um apanhado das matérias que deveriam ser melhoradas no *iLearnTest*, são elas:

- Pergunta 1 - Técnicas de teste estáticas;
- Pergunta 7 - Quais as técnicas de caixa-preta;
- Pergunta 9 - O que faz um testador de caixa-preta;
- Pergunta 17 - Exercício sobre cobertura de instruções;
- Pergunta 18 - Exercício sobre cobertura de decisões;
- Pergunta 19 - Quais as técnicas de caixa-branca;
- Pergunta 20 - Exercício sobre cobertura de instruções e decisões;
- Pergunta 21 - Exercício sobre cobertura de decisões.

Posto isto, foi realizada uma melhoria ao ensino do jogo. Criou-se uma página capaz de mostrar as diferenças entre cobertura de instruções, decisões e condições. Nela está presente uma função e desenhado o respetivo grafo de fluxo de controlo, o utilizador tem à disposíção três botões em que pode clicar que lhe mostram o que deve cobrir em testes de cobertura de intruções, decisões ou condições. Também se inseriu um jogo capaz de ensinar a distinção entre técnicas de caixa-preta e caixa-branca, em que o jogador deve arrastar as técnicas para dentro da respetiva caixa. Desta parte da experiência também resultou a descoberta de alguns *bugs* que também foram corrigidos.

Depois disto, como já foi referido na secção anterior, realizou-se a segunda parte da experiência. A tabela da figura 4.2, que apresenta o mesmo esquema de cores que a tabela da figura 4.1 já descrito anteriormente, demonstra a comparação de resultados, desta vez, entre o grupo sem *iLearnTest* (grupo A) e o grupo com o *iLearnTest* melhorado (grupo C).

Desta vez os resultados demonstraram-se favoráveis ao *iLearnTest*. O grupo C, que apresenta a média mais inferior dos grupos (13,33 valores), conseguiu atingir a melhor média de resultados no exame, 16,54 valores. Este grupo apresenta uma média geral de curso de menos 2,38 valores que o grupo A.

Após a obtenção destes resultados favoráveis, realizou-se uma análise mais aprofundada dos resultados de forma a entender o nível de aprendizagem que o *iLearnTest* é capaz de transmitir.

O gráfico da figura 4.3 mostra a relação entre as médias de curso (eixo das abcissas) dos quatro elementos do grupo A, que não utilizou o *iLearnTest*, e as notas obtidas no exame realizado na experiência (eixo das ordenadas). Neste gráfico observa-se uma linha de tendência com uma inclinação aproximada de 1,15 valores. Também é evidente que um aluno com média de 13,5 valores (valor aproximado à média das médias do grupo C), segundo esta tendência, obteria um

Grupo	Aluno	Média	Perguntas 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Resultado	Nota
A	1	15,71	1	1	1	1	1	0	1	0	1	1	0	1	1	1	1	1	1	0	1	0	1	16	16,80
	2		1	0	0	0	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	16	16,80
	3		1	1	0	1	1	1	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	17	17,85
	4		0	1	0	1	0	0	1	0	1	0	1	1	1	0	0	1	1	1	1	1	1	13	13,65
		Total:	3	3	1	3	3	2	4	1	3	2	3	4	3	2	3	4	4	3	4	3	4	15,50	16,28
C	9	13,33	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	17	17,85
	10		1	1	1	0	1	1	1	1	1	0	1	0	0	0	1	1	1	1	1	1	1	16	16,80
	11		1	1	0	1	1	0	1	1	0	1	1	1	1	1	0	1	1	1	1	0	0	15	15,75
	12		1	1	0	0	1	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	15	15,75
		Total:	3	4	1	1	4	2	4	3	2	3	4	3	3	2	3	4	4	4	4	2	3	15,75	16,54
		Total dos Grupos:	6	7	2	4	7	4	8	4	5	5	7	7	6	4	6	8	8	7	8	5	7		

Legenda de cores:
- Melhor resultado com iLT
- Melhor resultado sem iLT
- Todos os elementos acertaram
- 1 Elemento errou
- 2 Elementos erraram

Figura 4.2: Tabela de resultados da segunda parte da experiência.

resultado de 13 valores, o que não se verificou nos resultados deste grupo com a utilização do *iLearnTest*.

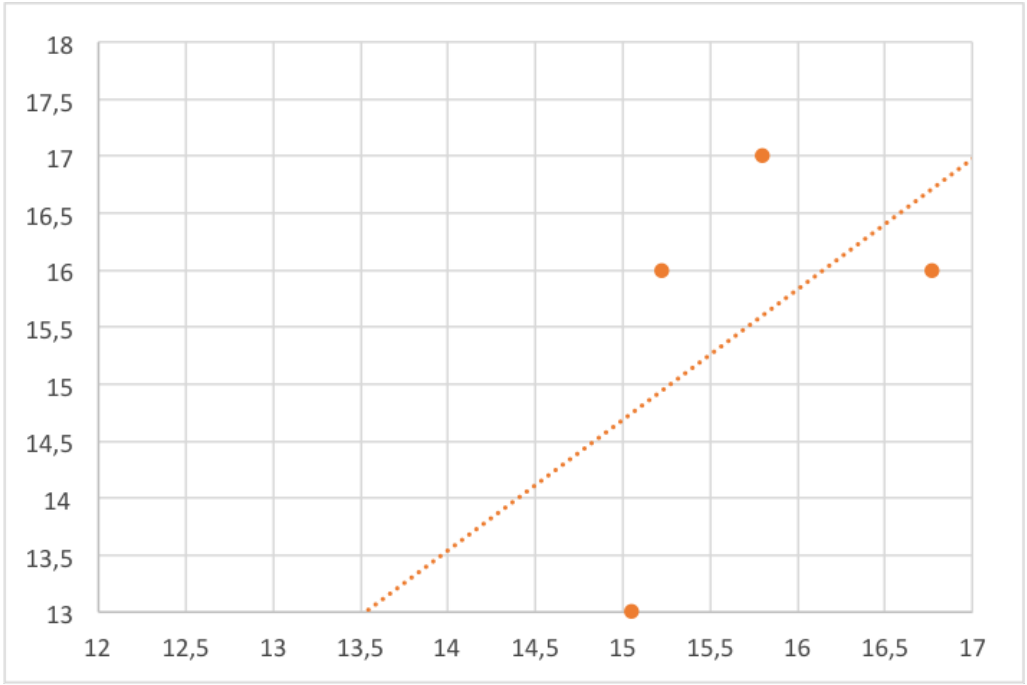


Figura 4.3: Gráfico de resultados da experiência sem *iLearnTest* (grupo A).

A linha de tendência do grupo que utilizou o *iLearnTest* apresenta uma inclinação de aproximadamente 0,54 valores, menor que a do grupo A (gráfico 4.3) como é possível observar pelo gráfico da figura 4.4. A diferença entre estas duas inclinações é aproximadamente de 0,6 valores. Segundo esta linha de tendência um aluno com uma média de 15,5 valores (média aproximada à média das médias do grupo A), se utiliza-se o *iLearnTest* para o seu estudo obteria um resultado

no exame de 17 valores, meio valor superior ao resultado médio obtido pelo grupo.

Por fim, o gráfico da figura 4.5 apresenta a relação entre as médias de curso (eixo das abcissas) dos elementos dos dois grupos da segunda parte da experiência (grupo A e C) e as notas obtidas no exame realizado na experiência (eixo das ordenadas). Os pontos cor de laranja presentes no gráfico identificam os alunos do grupo A. Já os pontos de cor azul identificam os alunos do grupo C. De acordo com a amostra de alunos utilizada, é possível observar pelas linhas de tendência desenhadas que os que utilizaram o *iLearnTest* têm tendência a obter melhores resultados.

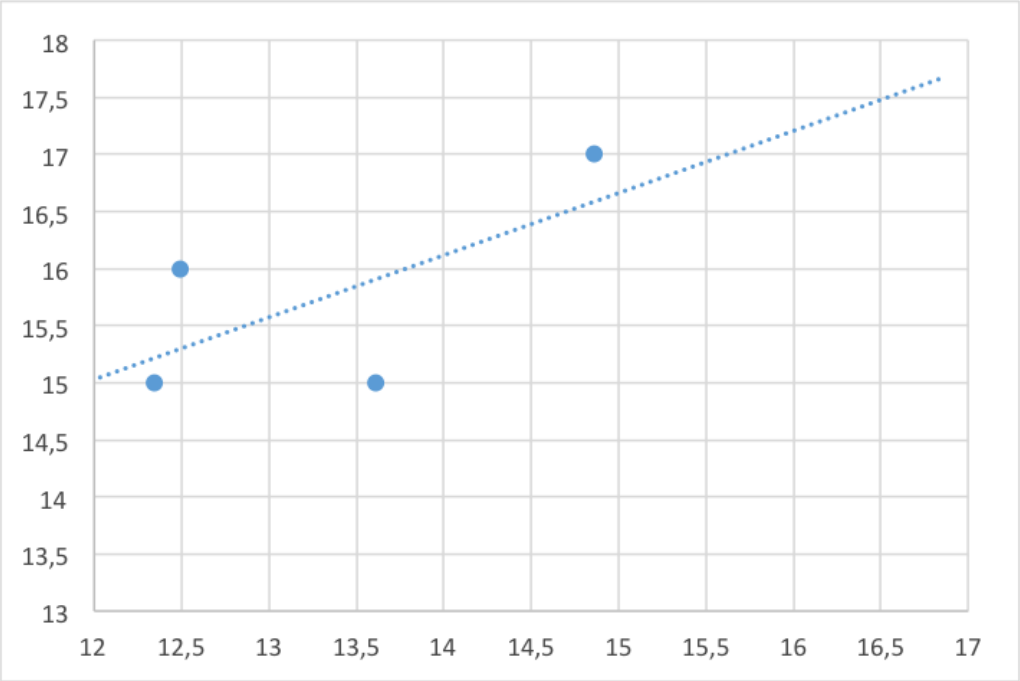


Figura 4.4: Gráfico de resultados da experiência com *iLearnTest* (grupo C).

Segundo a projeção da linha de tendência do grupo que não utilizou o *iLearnTest*, um aluno com média de curso de 12,5 valores obteria 12 valores no exame, com a utilização do jogo o mesmo aluno obteria 16 valores no exame, sendo um aumento de quatro valores. Já um aluno com média de 15,5 valores sem a plataforma obteria 15 valores no exame, e com ela obteria 17 valores, causando um aumento de dois valores. Com esta projeção podemos estimar que alunos com médias inferiores podem ter notas do exame superiores aqueles que estudaram sem a plataforma e que apresentam médias superiores.

4.5 Resumo e Conclusões

Este capítulo descreve a realização de uma experiência na qual se pretendia avaliar a eficácia do *iLearnTest* na transmissão dos conhecimentos de testes de software. Assim como a análise realizada aos seus resultados.

A validação do *iLearnTest* foi positiva e demonstrou que o jogo é eficaz na transmissão de conhecimentos de testes de software para a obtenção da certificação *Foundation* do ISTQB. A

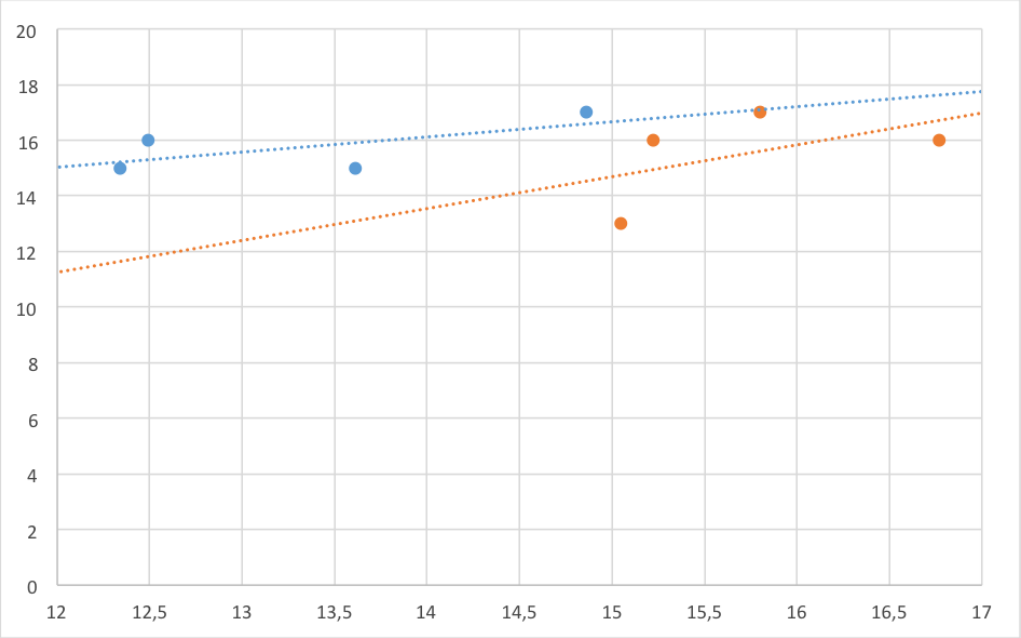


Figura 4.5: Gráfico de resultados totais da experiência.

utilização deste novo método de estudo mostrou-se até mais eficiente que a mera utilização do *Syllabus*, como demonstra a análise do gráfico da figura 4.5. Com esta experiência também se verificou que os estudantes se sentem mais motivados a estudar. A experiência foi realizada com grupos de quatro alunos mas para resultados mais fiáveis deveriam ter sido utilizados grupos maiores. Tornou-se complicado encontrar estudantes com tempo livre para participarem na experiência. Apesar disto, pode-se afirmar, segundo os resultados obtidos, que o *iLearnTest* criou uma forma mais aliciante e divertida de aprendizagem de testes de software.

Capítulo 5

Conclusões e Trabalho Futuro

Motivar e envolver os alunos como forma de melhorar a eficácia da aprendizagem tem sido alvo de estudo por parte de alguns investigadores, que sugerem para tal o uso de jogos sérios nas salas de aula [2]. Estudos anteriores, também, revelam que o uso de jogos como um complemento à tradicional aprendizagem é muito mais eficiente do que apenas a utilização do tradicional método de ensino [5, 6, 7, 8, 9, 10].

Atualmente, pouco software contempla a área de testes de software, constatando a real necessidade de criação do *iLearnTest*. Nesse contexto, este trabalho apresentou, o *iLearnTest*, um jogo sério capaz de apoiar o processo de ensino-aprendizagem de testes de software, com objetivo ensinar temas do corpo de conhecimento exigido pelo nível base da certificação do ISTQB.

O *iLearnTest* é uma contribuição divertida e motivacional no ensino de testes de software, com o propósito de servir como ferramenta de estudo para quem pretende obter a certificação *Foundation* do ISTQB. A aprendizagem transmitida baseia-se no programa *Syllabus* apresentado pelo ISTQB para a realização desta certificação. Como tal são transmitidos os mesmos objetivos de aprendizagem aos utilizadores.

No capítulo 1 é feita a introdução do trabalho, apresenta-se o enquadramento do trabalho, o projeto, a motivação e os objetivos do projeto a realizar e, por fim, descreve-se a estrutura do presente documento.

Para além da introdução, este documento contém mais quatro capítulos. No capítulo 2, é descrito o estado da arte e são apresentados projetos existentes na literatura dentro do mesmo domínio, e seus problemas, relevantes ao contexto do problema. Nele é também efetuada uma revisão tecnológica às principais ferramentas que poderão ser utilizadas no desenvolvimento do projeto. Em conclusão ao capítulo é apresentada a abordagem desenhada para a resolução do problema, em específico a utilização da metodologia ADDIE para o processo de desenvolvimento e a plataforma Construct 2 como ferramenta de implementação.

No capítulo 3 é apresentado o *iLearnTest*, os detalhes da sua implementação, os objetivos da aprendizagem que se encontram cobertos pela versão atual, o *website* construído e a descrição da

estrutura do jogo. Nele são também descritos os diferentes jogos/desafios implementados:

- *3.5.1 Adivinhar os Conceitos* - o jogador deve adivinhar os conceitos que são descritos pelos pontos apresentados;
- *3.5.2 Drag-and-drop para Duas Caixas* - o jogador deve organizar os conceitos apresentados no ecrã em duas caixas, de acordo com a sua categoria;
- *3.5.3 Apanhar os Conceitos Corretos* - o jogador tem de manipular o personagem para apanhar os conceitos corretos, deixando cair os incorretos;
- *3.5.4 Descobrir os Caminhos* - o jogador deve selecionar o caminho correto de um grafo de acordo com o pedido inicialmente.

A descrição da validação realizada ao projeto é feita no capítulo 4. Esta validação foi feita através de uma experiência com doze alunos, separados em três grupos, onde se fez uma comparação dos resultados obtidos pelos grupos num exame a que foram submetidos (do tipo de exame de obtenção da certificação do ISTQB), entre os grupos que utilizaram o jogo para o estudo e o grupo que não utilizou.

Por fim, este capítulo 5 apresenta as conclusões desta dissertação, a satisfação dos objetivos (secção 5.1) e o trabalho que poderá ser realizado no futuro (secção 5.2).

5.1 Satisfação dos Objetivos

Durante a realização da experiência, na qual se pretendia avaliar a eficácia do *iLearnTest* na transmissão dos conhecimentos de testes de software, foi possível verificar que os alunos que usaram o *iLearnTest* estavam mais motivados e divertidos.

Através da análise dos resultados da validação conclui-se que se desenvolveu uma nova ferramenta de ensino, baseada em aprendizagem por meio de jogos, eficaz no ensino de testes de software e com capacidade de motivar e entreter os utilizadores durante o processo de estudo, que muitas vezes pelo método tradicional se torna bastante monótono e aborrecido.

O *iLearnTest*:

- Incorpora desafios em cada matéria capazes de promover o envolvimento dos estudantes durante a prática dos conhecimentos e técnicas de teste de software;
- Oferece a possibilidade de estudo individual para que os alunos possam aprender ao seu próprio ritmo, gastando o tempo que achem necessário para a aprendizagem;
- Atribuí uma pontuação em cada jogo concluído pelo estudante, de forma a incentivar a obtenção da pontuação máxima e, por conseguinte, aumentar o seu conhecimento;
- Apresenta os resultados dos estudantes após cada desafio, indicando as respostas certas e erradas e apresentando a correção neste último caso.

Aliado a estas características, conseguiu-se, também, que o software esteja disponível através de um *browser* para um acesso simples, em qualquer lugar, e disponível a qualquer um em qualquer altura.

Em suma, podemos concluir, com a análise das características implementadas, com a experiência e com a análise dos seus resultados, que o *iLearnTest* é um jogo divertido que contribui para elevar a motivação dos alunos na aprendizagem de testes de software de acordo com o corpo de conhecimento do ISTQB. E, portanto, os objetivos deste projeto foram alcançados.

5.2 Trabalho Futuro

Até então o *iLearnTest* encontra-se na primeira versão. Numa futura versão do jogo, pretende-se cobrir os restantes conteúdos do *Syllabus* do ISTQB. Através da análise dos resultados da experiência realizada foi possível perceber quais os conteúdos que poderiam ser melhor explorados para aprimorar o *iLearnTest*. Assim, como forma de melhorar o que já está feito e o que poderá vir a ser feito, poderão ser utilizados os resultados já obtidos e, também, serem realizadas novas experiências com novos alunos, em trabalho futuro.

Bibliografia

[1] K. Kapp. *The gamification of learning and instruction: game-based methods and strategies for training and education*. United States of America: Pfeiffer; ASTD, 2012, p. 302.

[2] J. Fernandes e S. Sousa. «PlayScrum - A Card Game to Learn the Scrum Agile Method». Em: *Second International Conference on Games and Virtual Worlds for Serious Applications*. 2010, pp. 52–59.

[3] B. Schneiderman. «Designing for Fun: Can we design user interfaces to be more fun?» Thesis. 2005.

[4] M. Thirty, A. Zoucas e A. Silva. «Empirical Study Upon Software Testing Learning With Support From Educational Game». Em: *Proceedings of the 23rd International Conference on Software Engineering and Knowledge Engineering*. 2011, pp. 481–484.

[5] D. Carrington, A. Baker e A. van der Hoek. «It's all in the game: teaching software process concepts». Em: *Proceedings of the 35th Frontiers in Education Conference*. 2005.

[6] P. Mandell-Striegnitz. «How to successfully use software project simulation for educating software project managers». Em: *Proceedings of the 31st ASEE/IEEE Frontiers in Education Conference*. 2001.

[7] A. Drappa e L. Jochen. «Simulation in software engineering training». Em: *Proceedings of the 22nd International Conference on Software Engineering*. 2000, pp. 199–208.

[8] E. Navarro, A. Baker e A. Hoek. «Teaching Software Engineering Using Simulation Games». Em: *Proceedings of the International Western Simulation Multiconference*. 2004.

[9] E. Navarro e A. van der Hoek. «Software Process Modeling for an Educational Software Engineering Simulation Game». Em: *Software Process Improv Pract*. 2005, pp. 311–325.

[10] G. Taran. «Using games in software engineering education to teach risk management». Em: *Proceedings of the 20th Conference on Software Engineering Education & Training*. 2007, pp. 211–220.

[11] R. Rocha e R. Araujo. «Metodologia de Design de Jogos Sérios para Treinamento: Ciclo de vida de criação, desenvolvimento e produção». Em: *Proceedings of SBGames*. 2013.

[12] H. Chandler. *Manual de Produção de Jogos Digitais*. Porto Alegre: Bookman, 2012.

[13] P. Schuytema. *Design de Games: Uma Abordagem Prática*. São Paulo: Editora Cengage Learning, 2008.

[14] LLC. Stencyl. *Stencyl*. Web Page. 2014. URL: www.stencyl.com.

[15] Scirra. *Construct 2*. Web Page. 2007. URL: www.scirra.com.

[16] Unity Technologies. *Unity 3D*. Web Page. 2014. URL: www.unity3d.com.

[17] GameSalad. *GameSalad*. Web Page. 2011. URL: www.gamesalad.com.

[18] F. Benitti e J. Molléri. «Utilização de um RPG no Ensino de Gerenciamento e Processo de Desenvolvimento de Software». Em: *Workshop sobre Educação em Computação*. 2008, pp. 258–267.

[19] Z. Qing, W. Tao e T. Shenglong. «Adapting Game Technology to Support Software Engineering Process Teaching: From SimSE to MO-SEProcess.» Em: *3rd International Conference on Natural Computation*. 2007, pp. 777–780.

[20] S. Elbaum et al. «Bug Hunt: Making Early SoftwareTesting Lessons Engaging and Affordable». Em: *29th International Conference on Software Engineering*. 2007, pp. 688–697.

[21] B. Oliveira. «TestEG - Um Software Educational para o Ensino de Teste de Software». Thesis. 2013.

[22] ISTQB. *Certified Tester, Foundation Level Syllabus*. Generic. 2011.

Anexo A

Exame da Experiência

Nas próximas seis páginas encontra-se anexado o exame que aos participantes realizaram no final da experiência montada com o objetivo de fazer a validação do iLearnTest.

Nele constam vinte e uma perguntas de escolha múltipla do mesmo estilo de perguntas do exame de certificação *Foundation* do ISTQB.

Questions about Software Testing

Group: _____

This is a test of 21 multiple-choice questions, whose format is similar to the ISTQB Certified Tester Foundation Level Exam. The objective of the test is to compare results from two distinct groups, one studied from ISTQB Syllabus Foundation and the other from iLearnTest.

1. Select the statement that completes the phrase "Static testing techniques…".

- ☐ should be always applied in conjunction with dynamic testing
- ☐ is a method of assessing the feasibility of a software program by giving input and examining output
- ☐ relies on the manual examination and automated analysis
- ☐ requires the execution of the software

2. Which of the following are the main phases of a formal review?

- ☐ Initiation, status, preparation, review meeting, rework, follow up
- ☐ Planning, preparation, review meeting, rework, closure, follow up
- ☐ Planning, kick off, individual preparation, review meeting, rework, follow up
- ☐ Preparation, review meeting, rework, closure, follow up, root cause analysis

3. Select the statement that completes the phrase "A manager in a formal review should…".

- ☐ Determine if the review objectives have been met
- ☐ Document all the issues
- ☐ Mediate between the various points of view
- ☐ Plan and run the review

4. Which of the following statements about static analysis is FALSE?

- ☐ Static analysis can be used as a preventive measure with appropriate process in place
- ☐ Static analysis can find defects that are not easily found by dynamic testing
- ☐ Static analysis can result in cost savings by finding defects early
- ☐ Static analysis is a good way to force failures into the software

5. Which of the following is a typical defect discovered by static analysis tools?

- ☐ Referencing a variable with a defined value
- ☐ Too much use of a variable
- ☐ Unreachable (dead) Code
- ☐ All of the above

6. Select the statement that completes the phrase "A test design technique is...".

- ☐ a process for determining expected outputs
- ☐ a process for selecting test cases
- ☐ a way to measure in a test plan what has to be done
- ☐ a way to measure the quality of software

7. Which of the following is a black box design technique?

- ☐ Equivalence partitioning
- ☐ Condition coverage
- ☐ Statement testing
- ☐ Path coverage

8. Which of the following techniques is NOT a black box design technique?

- ☐ Boundary value analysis
- ☐ LCSAJ
- ☐ State transition testing
- ☐ Syntax testing

9. Select the statement that do NOT completes the phrase "The black box tester...".

- ☐ is creative to find the system's weaknesses
- ☐ is highly motivated to find faults
- ☐ should be able to understand a functional specification or requirements document
- ☐ should be able to understand the source code

10. Which of the following solutions below lists techniques that can all be categorized as black box design techniques?

- ☐ Equivalence Partitioning, cause-effect graph, checklist based, decision coverage, use case
- ☐ Equivalence Partitioning, cause-effect graph, checklist based, decision coverage, boundary value
- ☐ Equivalence Partitioning, decision table, checklist based, statement coverage, use case
- ☐ Equivalence Partitioning, decision table, state transition, boundary value

11. Which of the following statements is TRUE for the equivalence partitioning test technique?

A. Divides possible inputs into classes that have the same behavior
B. Uses both valid and invalid partitions
C. Makes use only of valid partitions
D. Must include at least two values from every equivalence partition
E. Can be used only for testing equivalence partitions inputs from a Graphical User Interface

- ☐ A and B are true; C, D and E are false
- ☐ A, B and E are true; C and D are false
- ☐ A, C and D are true; B and E are false
- ☐ A and E are true; B, C and D are false

12. A program validates a numeric field as follows:
"Values less than 10 are rejected, values between 10 and 21 are accepted, values greater than or equal to 22 are rejected."
Which of the following input values cover all of the equivalence partitions?

- ☐ 3, 10, 22
- ☐ 3, 20, 21
- ☐ 10, 11, 21
- ☐ 10, 21, 22

13. An employee's bonus is to be calculated. It cannot become negative, but it can be calculated to zero. The bonus is based on the duration of the employment. An employee can be employed for less than or equal to 2 years, more than 2 years but less than 5 years, 5 to 10 years, or longer than 10 years. Depending on this period of employment, an employee will get either no bonus or a bonus of 10%, 25% or 35%.
How many valid equivalence partitions are needed to test the calculation of the bonus?

- ☐ 3
- ☐ 2
- ☐ 4
- ☐ 5

14. Select the statement that do NOT completes the phrase "A Use Case...".

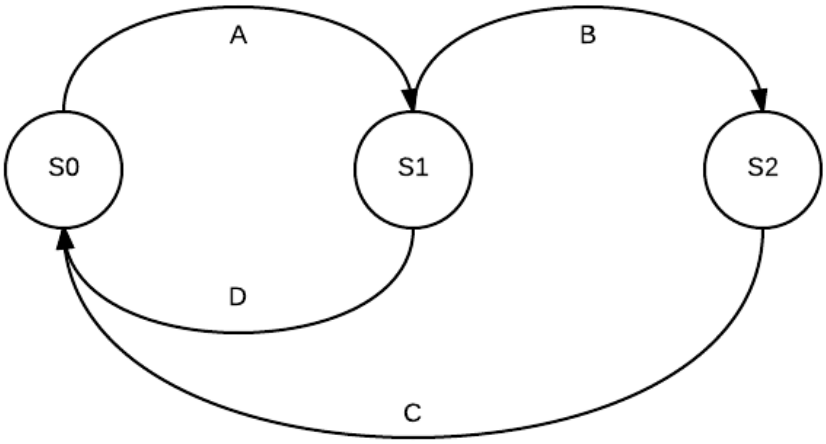
- ☐ describes interaction between actors
- ☐ may be described at the business abstract level
- ☐ may be described at the system functionality level
- ☐ shouldn't be combined with other back box techniques

15. Given the following decision table: Which of the following test cases and expected results is VALID?

	Rule 1	Rule 2	Rule 3	Rule 4
Conditions				
Age	<21 yrs	21-29 yrs	30-50yrs	> 50yrs
Insurance Class	A	A or B	B. C or D	C or D
Actions				
Premium	£100	£90	£70	£70
Excess	£2,500	£2,500	£500	£1000

- ☐ 23 year old in insurance class A Premium is 90 and excess is 2,500
- ☐ 31 year old in insurance class B Premium is 90 and excess is 1,500
- ☐ 43 year old in insurance class C Premium is 70 and excess is 1,000
- ☐ 51 year old in insurance class C Premium is 90 and excess is 1,000

16. Given the following state transition table: Which of the test cases below will cover the following series of state transitions, S1 SO S1 S2 SO?



- ☐ A, B, C
- ☐ A, B, C, D
- ☐ D, A, B
- ☐ D, A, B, C

17. If the pseudo code below were a programming language, how many tests are required to achieve 100% statement coverage?

```
If x = 3 then
    Display_messageX;
    If y = 2 then
        Display_messageY;
    Else
        Display_messageZ;
Else
    Display_messageZ;
```

- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4

18. If the pseudo code below were a programming language, how many tests are required to achieve 100% decision coverage?

```
If x = 3 then
    Display_messageX;
    If y = 2 then
        Display_messageY;
    Else
        Display_messageZ;
Else
    Display_messageZ;
```

- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4

19. Select the statement that completes the phrase "Condition coverage and path coverage are part of…".

- ☐ Black box testing
- ☐ Life cycle testing
- ☐ Regression testing
- ☐ White box testing

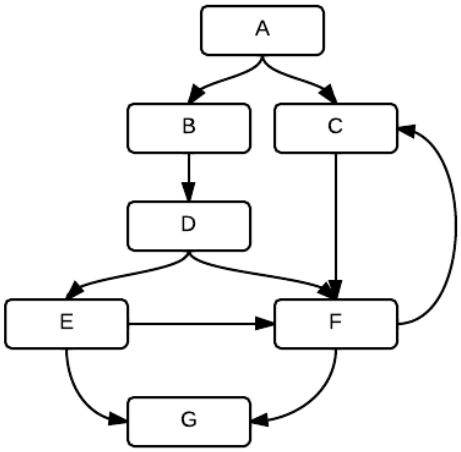
20. Given the following code, which statement is true about the minimum number of test cases required for full statement and decision coverage?

```
Read p
Read q
If p + q > 100 then
    Print "Large"
Endif
If p > 50 then
    Print "p Large"
Endif
```

- ☐ 1 test for statement coverage, 1 for decision coverage
- ☐ 1 test for statement coverage, 2 for decision coverage
- ☐ 1 test for statement coverage, 3 for decision coverage
- ☐ 2 tests for statement coverage, 2 for decision coverage

21. One of the test goals for the project is to have 100% decision coverage. The following three tests have been executed for the control flow graph shown below. Which of the following statements related to the decision coverage goal is correct?

Test A covers path: A, B, D, E, G
Test B covers path: A, B, D, E, F, G
Test C covers path: A, C, F, C, F, C, F, G



- ☐ Decision D has not been tested completely
- ☐ Decision E has not been tested completely
- ☐ Decision F has not been tested completely
- ☐ 100% decision coverage has been achieved

Anexo B

Imagens do *website* do *iLearnTest*

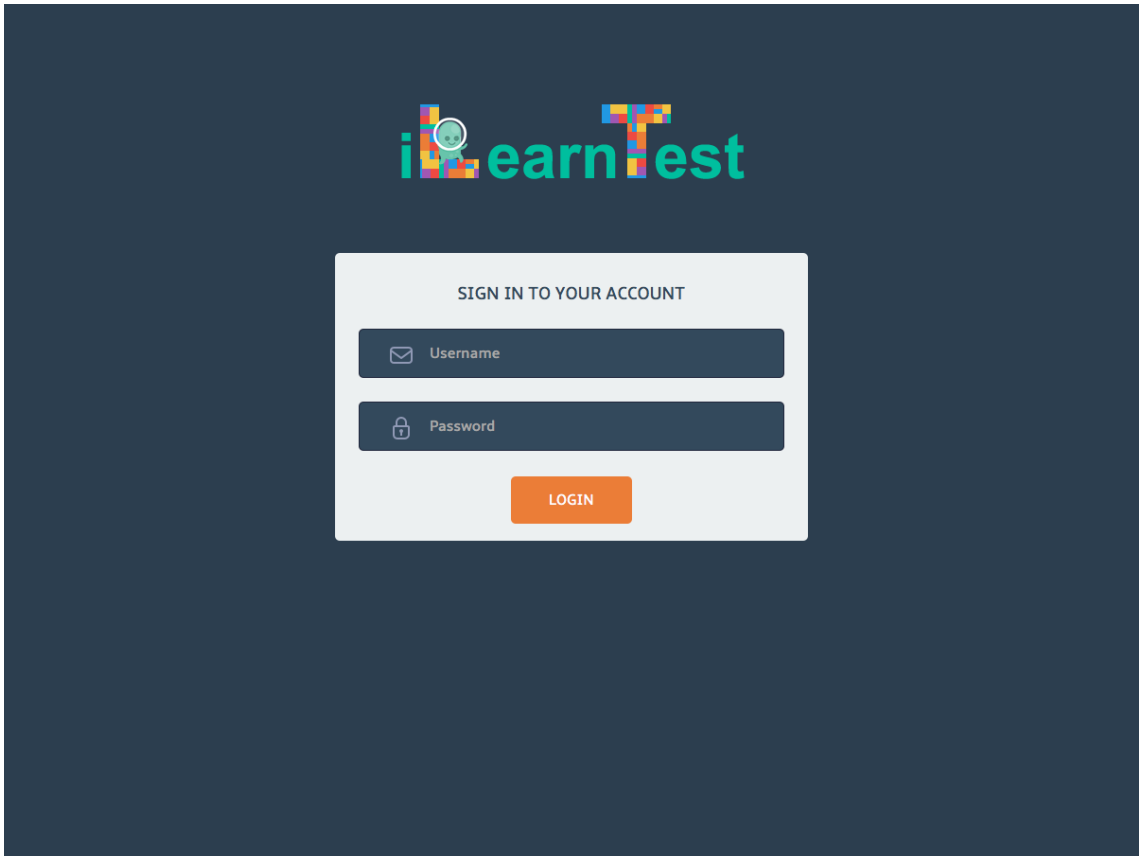


Figura B.1: Imagem da página de autenticação do *website* do *iLearnTest*.

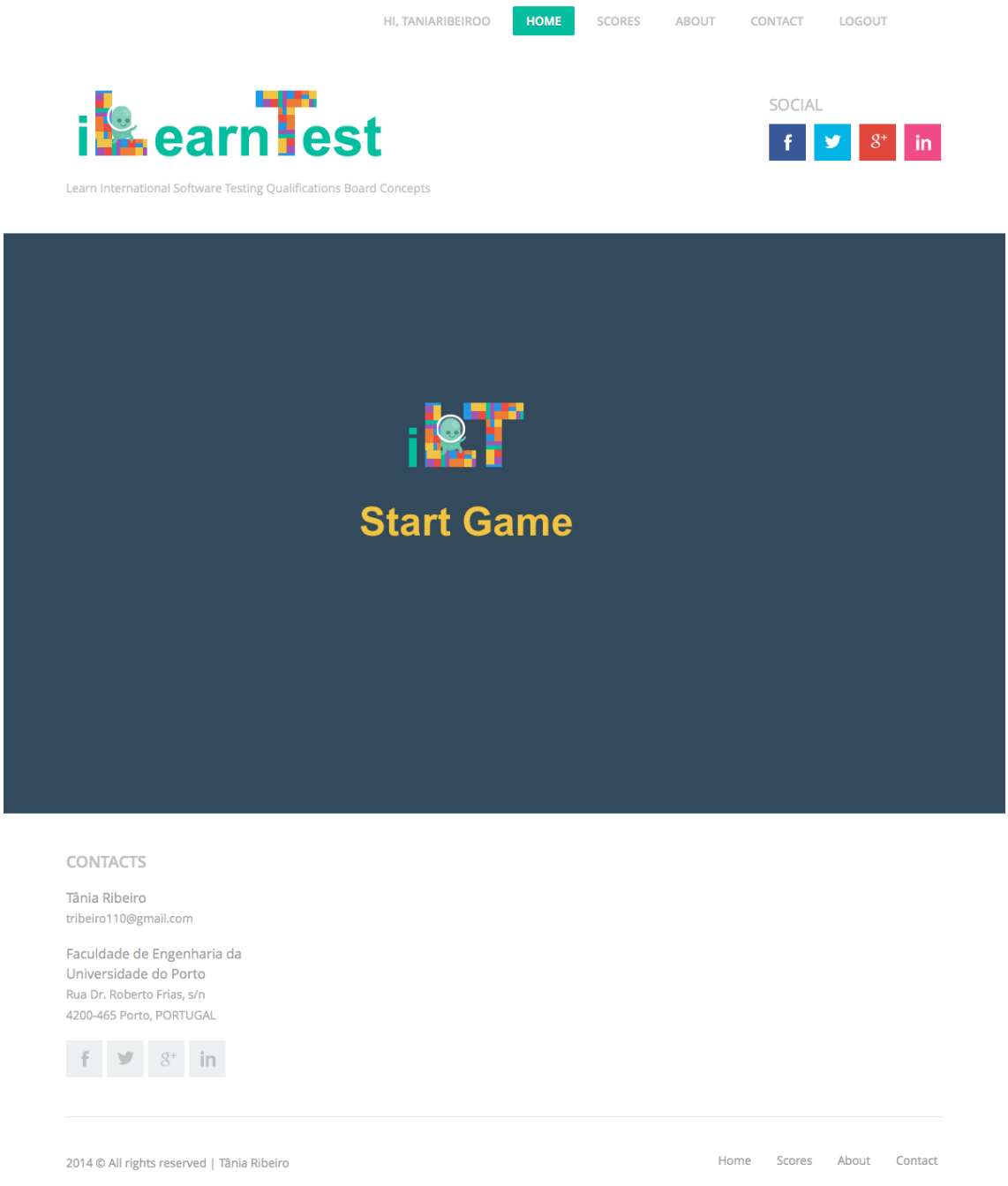


Figura B.2: Imagem da página inicial do *website* do *iLearnTest*.

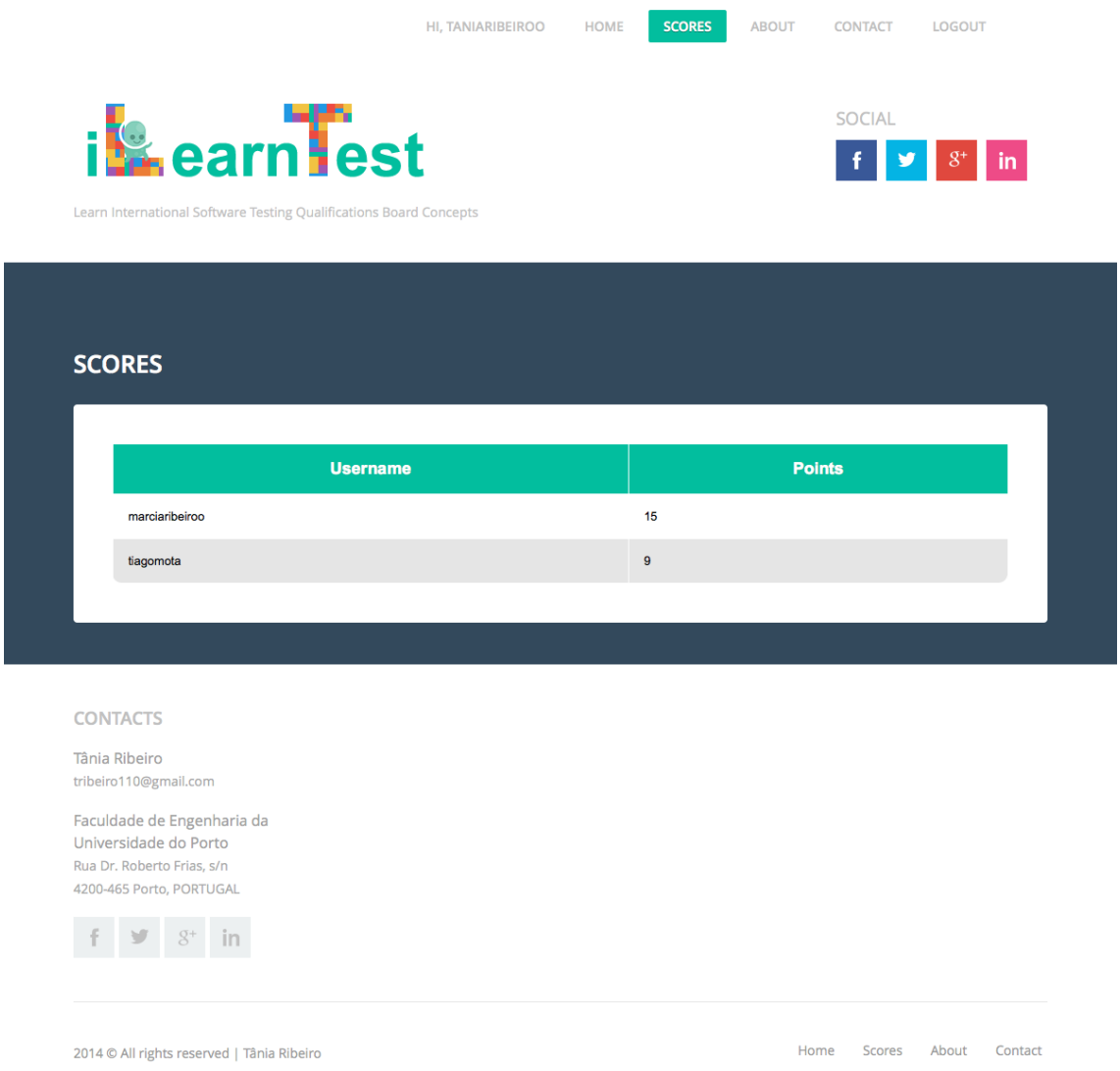


Figura B.3: Imagem da página de pontuações do *website* do *iLearnTest*.